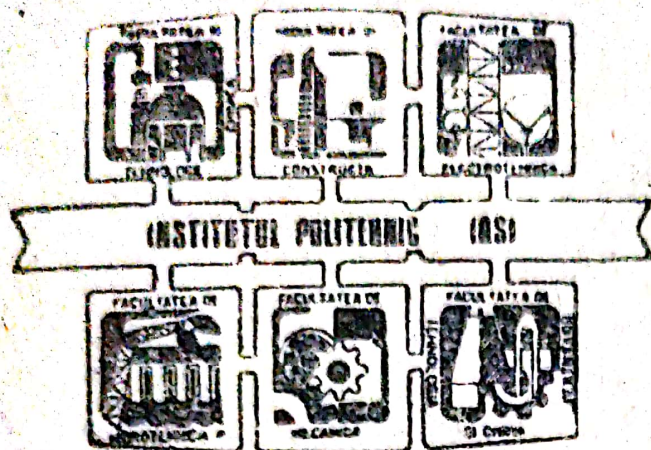


INSTITUTUL POLITEHNIC IAȘI
FACULTATEA DE TEHNOLOGIA ȘI CHIMIA TEXTILELOR



Mihai Ciocoiu

**PROGRAMAREA
LA
CALCULATOARELE ELECTRONICE**

(Inițiere în limbajul FORTRAN)

***Pentru uzul studenților
IAȘI — 1985***

I N S T I T U T U L P O L I T E C N I C I A S I
F A C U L T A T E A D E T E C N O L O G I A S I C H I M I A T E X T I L E L O R

M I H A I C I O C O I U

P R O G R A M A R E A
L A
C A L C U L A T O A R E L E E L E C T R O N I C E

(Inițiere în limbajul FORTRAN)

I A S I - 1985

I N T R O D U C E R E

Componentă a pachetului de discipline de cultură tehnică generală din planul de învățămînt, disciplina de "Programarea la calculatoarele electronice" are rolul de a pune la îndemîna viitorului inginer textilist cunoștințele necesare referitoare la utilizarea calculatorului electronic în rezolvarea unor probleme cu caracter tehnico-științific. De aceea manualul prezintă în primul capitol elemente ale bazelor teoretice ale construcției și funcționării calculatoarelor.

În următorul capitol - al 2-lea - sînt prezentate schemele logice însoțite de exemple pentru o mai bună înțelegere a noțiunilor.

În capitolele următoare este prezentat limbajul de programare FORTRAN.

Ordinea așezării capitolelor a fost astfel aleasă încît să corespundă etapelor necesare parcurgerii limbajului. Sperăm că astfel se oferă studentului o mai facilă asimilare a cunoștințelor. Fiind un curs, numărul exercițiilor prezentate este minim. Aceasta pentru a evita consumul de hîrtie și pentru că studenții textiliști au la dispoziție trei culegeri de probleme publicate de autor în colaborare în anii precedenți.

Apelarea la aceste culegeri, și nu numai la ele, este absolut necesară pentru buna înțelegere a celor prezentate în prezentul manual. O bibliografie selectată de autor, folosită de altfel pentru redactarea cursului de față, însoțește cursul.

Autorul dorește să exprime, încă odată, mulțumirile sale comisiei de analiză formată din conf.dr.mat.Gh.Ciobanu, șef lucr.dr.ing.mat.N.Cojocaru de la I.P.Iași și lector dr.mat.Gh.Grigoraș de la Universitatea "Al.I.Cuza" Iași pentru prețioasele îndrumări și observații făcute. Deasemănă dorește să mulțumească colegilor de la Centrul de calcul al I.P.Iași care, citind cu atenție manuscrisul, i-au făcut numeroase sugestii folositoare.

Convins de adevărul afirmației "patru ochi văd mai bine
ca doi" și de faptul că se pot aduce îmbunătățiri conținutu-
lui prezentului manual, autorul mulțumește anticipat tuturor
celor care vor veni cu sugestii și propuneri în acest sens.

Dr.ing.mat. Mihai Ciocoin

CAPITOLUL 1

BAZELE TEORETICE ALE CALCULATOARELOR

1.1. Scurt istoric al calculatoarelor

Din cele mai vechi timpuri omul a fost preocupat de construcția diferitelor unelte care să-i ușureze munca sau care să-l ajute să efectueze diferite activități mai rapid. Una din aceste activități a fost și activitatea de efectuare a calculului.

Unul din cele mai simple și totodată mai vechi instrumente de calcul este abacul, utilizat până în timpurile noastre. Trecea la sistemul de numerație arab, prin secolul XII, a dus la dezvoltarea rapidă a științei în general și a matematicii în special. Astfel în 1617 John Napier introduce noțiunea de logaritm, în baza căreia în 1621 William Oughtred a realizat rigla de calcul, instrument de mare ajutor pentru ingineri și folosit cu succes până acum 15-20 ani. La începutul anilor 1600, Blaise Pascal, pe vremea aceea în vîrstă de numai 18 ani, construiește pentru tatăl său, administrator de impozite la Rouen, o mașină mecanică de calculat care a avut mare succes. În jurul anilor 1675, G.Leibniz imaginează mașina de calculat care realizează înmulțirea prin adunări repetate. Au urmat diferite perfecționări aduse de C.X. Thomas, D.D.Parmelle etc. În jurul anilor 1800, M.Jacard inventează o mașină de țesut la care comanda mașinii este stocată pe cartele de carton, printr-o succesiune de perforații, metodă utilizată și astăzi la introducerea datelor în calculator.

În 1820 Charles Babbage imaginează o mașină de calcul prevăzută să memoreze comenzile necesare soluționării problemei, să efectueze calculele, să memoreze rezultatele, apoi să le furnizeze celui interesat.

La sfîrșitul secolului XIX Hollerith realizează cartela perforată care îi poartă numele și care este unul din suportii de informație utilizați cel mai mult în calculatoare.

Ideea lui Babbage a fost realizată în 1944, cînd la firma IBM (International Business Machines Corporation) se construiește mașina de calcul MARK I. Era o mașină de calcul cu rețea electro-



magnetice. Nu era încă un calculator electronic.

În 1946 apare primul calculator electronic denumit ENIAC (Electronic Numerical Integrator and Calculator). Era o mașină ocupând 160 m^2 , având în componența sa 20 000 tuburi. Avea o viteză de lucru de 5000 operații pe secundă. S-a născut generația întâia de calculatoare, construite cu tuburi electronice, caracterizate de viteză relativ redusă (de ordinul a 10^3 operații pe secundă), memorie redusă, programabile în limbaj-mașină. Au deschis era informaticii științifice, când calculatoarele erau folosite numai în domeniul cercetării.

În jurul anilor 1955-1956 apare generația a doua de calculatoare, la care se foloseau în locul tuburilor electronice tranzistoarele și diodele. Memoria internă se realizează pe ferite.

Ca urmare, volumul acestor calculatoare a scăzut, a sporit capacitatea de memorare, a crescut spectaculos viteza (100 000 operații pe sec.). Apar limbajele universale de programare. Au apărut sistemele de operare. Calculatoarele trec în domeniul activității economice - informatica de gestiune.

Generația a treia de calculatoare a apărut în jurul anului 1965 și folosește circuitele integrate. Gabaritul a scăzut mult, viteza de lucru a ajuns la 10^6 operații pe secundă. Calculatorul își lărgeste aria de utilizare, intrând în faza informaticii de organizare a întreprinderii.

În jurul anilor 1970 apar calculatoare din generația 3,5 - 4, care folosesc circuite integrate pe scară largă (LSI).

În anii 1985 - 1990 se așteaptă să devină operaționale calculatoarele cu viteze de lucru de ordinul miliardelor de operații pe secundă. Ele vor folosi principii noi de lucru, cum ar fi: laseri, fibra de sticlă, criogenia, tehnica înălțurii în rețele unice a sute și mii de calculatoare.

În anii ce vin microprocesorul va înlocui actualele sisteme de calcul datorită avantajelor pe care le prezintă, și anume: dimensiuni reduse, fiabilitate în funcționare. De exemplu, un microprocesor de câteva grame are aceleași posibilități ca un calculator de generația întâia, care cântărea 30 t.

La noi în țară s-au construit calculatoare electronice încă din 1957 la IFA-București, denumite CIFA, la Timișoara-MECIPT, la Cluj-DACCIC, calculatoare din generația întâia. Au urmat calculatoarele CET-501, DACCIC-202 din generația a doua.

În anul 1967 a apărut Hotărîrea C.C. al P.C.R. privind dotarea cu tehnică de calcul. Hotărîrile din 1972 și din Direcți-

vele Congreselor Partidului XI și XII au urmărit cu perseverență amplificarea acțiunii de dotare cu tehnică de calcul, dezvoltarea cercetării în domeniul tehnicii de calcul, extinderea domeniilor de aplicare a tehnicii de calcul, înființarea unor întreprinderi ce produc elemente componente necesare tehnicii de calcul.

Din 1970, la noi în țară se fabrică calculatoare din generația a treia, din familia FELIX. În momentul de față se fabrică calculatoare din generația 3 - 3,5 din familia FELIX (FELIX 512/1024); minicalculatoare CORAL 4001/4011, INDEPENDENT 100 și altele.

În toate județele țării, începând cu anul 1970, sînt organizate centre teritoriale de calcul. Toate întreprinderile, în funcție de mărimea acestora, sînt prevăzute cu oficii sau centre de calcul.

S-a trecut în ultimii ani la interconectarea centrelor teritoriale de calcul, la nivel național - rețeaua rezultată este denumită simbolic UNIREA. Conform Directivelor Congresului al XIII-lea se va realiza sistemul informatic național, se vor realiza programe eficiente, tipizate, generalizabile la cît mai mulți beneficiari, se va perfecționa modul de utilizare a tehnicii de calcul, se vor organiza baze și bănci de date conform standardelor internaționale, se va dezvolta robotica, etc.

Introducerea informaticii în organizarea și desfășurarea activităților celor mai diverse din societate este un proces dialectic, ireversibil.

1.2. Clasificarea calculatoarelor

Calculatoarele electronice se clasifică după tipul informațiilor prelucrate, după domeniul de utilizare și după modul de funcționare (fig. 1.1.)

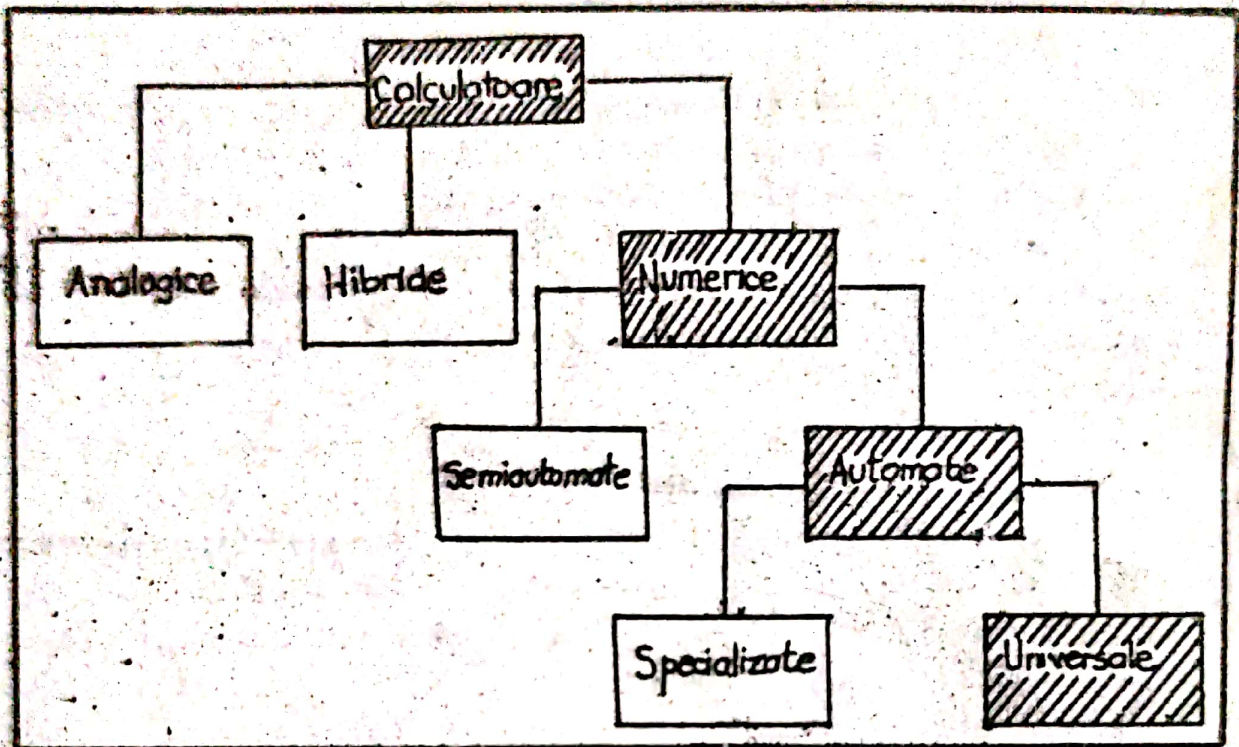


Fig. 1.1 Clasificarea calculatoarelor

Calculatoarele analogice prelucrează mărimi de tip continuu, variația acestora fiind reprezentată prin analogie, cu variația tensiunilor în elementele electronice de calcul, cu o precizie redusă (cele mai perfecționate: 0,1 %), iar domeniul lor de utilizare este restrâns.

Calculatoarele numerice prelucrează mărimi de tip discret.

Ideea construirii calculatoarelor numerice a pornit de la analiza efectuării calculului de către un operator uman, stabilind apoi mijloacele de automatizare ale procesului de calcul.

Prelucrarea datelor de către calculatorul electronic numeric se face cu o viteză mare de calcul, de aceea controlul efectuării operațiilor poate fi automat sau semiautomat.

Deoarece domeniile de utilizare ale calculatoarelor sunt extrem de diverse este evident că utilizatorii vor dori calculatorul care să răspundă unor anumite cerințe, numite calculatoare specializate, fie calculatoare utilizabile într-o foarte

largă arie de probleme - calculatoare universale.

În cursul de față se vor prezenta aspectele legate de utilizarea calculatoarelor electronice numerice, automate, universale.

Trebuie subliniat că în literatură apar des, alături de noțiunile de calculator, noțiunile de ordinador, computer sau, în ultimul timp, sistem de prelucrare automată a datelor (S.P.A.D.) sau sistem automat de calcul (S.A.C.).

1.3. Bazele aritmetice ale calculatoarelor

1.3.1. Sisteme de numerație

Datele numerice se pot reprezenta în diferite sisteme de numerație, un astfel de sistem fiind caracterizat prin numărul de simboluri permise pentru reprezentarea unui număr.

Se observă că un număr scris în sistemul de numerație zecimal, cu care sîntem atît de bine familiarizați, spre exemplu: $N = 9163$ reprezintă în realitate suma $9 \cdot 10^3 + 1 \cdot 10^2 + 6 \cdot 10^1 + 3 \cdot 10^0$ (unde $10^0 = 1$). Deoarece numerele zecimale sînt dezvoltate în puteri ale lui 10, se spune că acest sistem are baza 10.

Baza sistemului de numerație indică numărul de simboluri folosite pentru reprezentarea unui număr în sistemul respectiv de numerație. În sistemul zecimal numărul de simboluri permise pentru o cifră este 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Cele expuse mai sus se pot generaliza pentru definirea sistemelor numerice cu orice bază reprezentată printr-un număr întreg ($b > 1$). Dacă numărul N , pozitiv, este scris în baza b , atunci poate fi dezvoltat astfel:

$$N = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b^1 + a_0 b^0 \text{ pentru } N \text{ întreg}$$

sau

$$N = a_{-1} b^{-1} + a_{-2} b^{-2} + \dots + a_{-n} b^{-n} \text{ pentru } 0 \leq N < 1$$

unde coeficienții $a_0, a_1, \dots, a_{-1}, a_{-2}, \dots, a_{-n}$ reprezintă cifrele numărului scris în baza b ; aceste cifre pot fi reprezentate prin b simboluri și anume: 0, 1, 2, ..., (b-1).

Un număr N , reprezentat în sistemul de numerație în baza b , se scrie convențional prin juxtapunerea (alăturarea) coeficienților a_i astfel:

$$N = a_n a_{n-1} a_{n-2} \dots a_2 a_1 a_0$$

Prin aceasta am reprezentat numărul

$$N = a_n b^n + \dots + a_1 b^1 + a_0 b^0$$

Un număr pozitiv care are și parte întreagă și parte fracționară se scrie în sistemul de numerație în baza b astfel:

$$N = a_r b^r + a_{r-1} b^{r-1} + \dots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \dots + a_{-n} b^{-n}$$

$$\text{sau } N = a_r a_{r-1} \dots a_1 a_0 a_{-1} \dots a_{-m}$$

unde coeficienții a_r se pot reprezenta prin b simboluri.

Numărul $N_{10} = 9426,392$ este reprezentat în sistem zecimal astfel :

$$N_{10} = 9 \cdot 10^3 + 4 \cdot 10^2 + 2 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 9 \cdot 10^{-2} + 2 \cdot 10^{-3}$$

Prezintă interes următoarele sisteme de numeratie în ce privește utilizarea lor în calculatoare :

Tabel 1.1.

Sistemul de numeratie	Baza	Nr. simboluri pt. cifre	Simboluri pentru cifre
Binar	2	2	0, 1
Octal	8	8	0, 1, 2, 3, 4, 5, 6, 7
Zecimal	10	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexazecimal	16	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Regula scrierii numerelor formate din mai mult de o cifră este aceeași pentru toate sistemele de numeratie și anume : orice număr se scrie ca un gir de cifre din sistemul de numeratie considerat avînd pe prima poziție întotdeauna o cifră diferită de zero (vezi tabelul 1.2).

1.3.2. Operații aritmetice în diferite sisteme de numeratie

Se vor prezenta pe scurt modul de efectuare a următoarelor operații : adunarea, scăderea, înmulțirea, împărțirea în sistemele de numeratie menționate.

În principiu adunarea și înmulțirea se efectuează pe baza unor tabele de operații întocmite în sistemul respectiv de numeratie, tabele în care se înregistrează rezultatul efectuării acelei operații între două numere de câte o cifră.

Aceste tabele se pot stabili pentru orice sistem de numeratie.

Tabla adunării binare

+	0	1
0	0	1
1	1	10

Tabla înmulțirii binare

x	0	1
0	0	0
1	0	1

Tabel 1.2.

Sistem desimal	Sistem binar	Sistem oktal	Sistem heksadesimal
(b=10)	(b=2)	(b=8)	(b=16)
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
100	1100100	144	64

Codul EBCDIC

Tabel 1.3.

Caractere					
Litere		Cifre		Semne speciale	
Litera	Codul EBCDIC	Cifra	Codul EBCDIC	Semnul	Codul EBCDIC
A	C1	0	F0	+	4E
B	C2	1	F1	-	60
C	C3	2	F2	(	4D
D	C4	3	F3	)	5D
E	C5	4	F4	.	6B
F	C6	5	F5	.	4B
G	C7	6	F6	-	7E
H	C8	7	F7		4C
I	C9	8	F8		6E
J	D1	9	F9	#	5C
K	D2			.	7D
L	D3			spatiu	40
M	D4			/	61
N	D5			\$	5B
O	D6			:	5E
P	D7			\$	50
Q	D8			%	6C
R	D9			?	6F
S	E2			:	7A
T	E3				
U	E4				
V	E5				
W	E6				
X	E7				
Y	E8				
Z	E9				

adunării octale :

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Tabla înmulțirii octale:

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	31
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Tabla adunării hexazecimale

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Similar există și o tablă a înmulțirii hexazecimale.

La adunare, cât și la scădere, în toate sistemele de nume-
reție "transportul", respectiv "împrumutul", de la poziția de
rang imediat superior se face exact ca în sistemul zecimal.

Sisteme	Exemple de adunare	Exemple de scădere
în sistem binar	$\begin{array}{r} 1101 + \\ 101 \\ \hline 10010 \end{array}$	$\begin{array}{r} 1101 - \\ 110 \\ \hline 0111 \end{array}$
în sistem octal	$\begin{array}{r} 635 + \\ 466 \\ \hline 1323 \end{array}$	$\begin{array}{r} 752 - \\ 65 \\ \hline 665 \end{array}$
în sistem hexazecimal	$\begin{array}{r} 2A9B + \\ 85A3 \\ \hline B03E \end{array}$	$\begin{array}{r} B03E - \\ 2A9B \\ \hline 85A3 \end{array}$

Observație : Prin " ← " s-a notat "transportul"

Prin " → " s-a notat "împrumutul"

Operațiile de înmulțire și împărțire în toate sistemele de numerație, respectă regulile cunoscute de la sistemul zecimal.

Sistem	Exemple de înmulțire	Exemple de împărțire
sistem binar	$\begin{array}{r} 1011 \times \\ 101 \\ \hline 1011 \\ 1811 \\ \hline 110111 \end{array}$	$\begin{array}{r} 110111 : 101 \\ - 101 \\ \hline 111 \\ - 101 \\ \hline 101 \\ - 101 \\ \hline 000 \end{array}$
sistem octal	$\begin{array}{r} 775 \times \\ 56 \\ \hline 5756 \\ 4761 \\ \hline 55566 \end{array}$	$\begin{array}{r} 55566 : 56 \\ - 502 \\ \hline 530 \\ - 502 \\ \hline 306 \\ 346 \\ \hline 000 \end{array}$
sistem hexazecimal	$7 \times 8 = 38$	$7E : E = 6$

Operația de bază într-un calculator numeric este operația de adunare (între numere reprezentate în sistemul de numerație binar) ; celelalte operații aritmetice se realizează pe baza unor reguli bine determinate (algoritme de calcul), prin adunări sau scăderi repetate.

1.3.5. Conversiunea numerelor

Prin conversiune se înțelege operația prin care se obține reprezentarea unui număr într-un sistem de numerație, pornind de la reprezentarea sa în alt sistem de numerație.

În calculatoarele electronice numerice datele se reprezintă în sistemul binar; uneori se preferă reprezentarea datelor binare în sistemul hexazecimal deoarece numărul de cifre componente ale unui număr este mai mic în acest sistem. Astfel, dacă reprezentăm numărul $N = 6033$ din zecimal în sistemul binar, octal, hexazecimal:

Sistem de numerație	Reprez.nr. N_{10}	Verificare
binar	1011110010001	$1 \cdot 2^{12} + 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^4 + 1 \cdot 2^0 = 6033$
octal	13621	$1 \cdot 8^4 + 3 \cdot 8^3 + 6 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0 = 6033$
zecimal	6033	$6 \cdot 10^3 + 3 \cdot 10^1 + 3 \cdot 10^0 = 6033$
hexazecimal	1791	$1 \cdot 16^3 + 7 \cdot 16^2 + 9 \cdot 16^1 + 1 \cdot 16^0 = 6033$

Conversiunea binar-zecimală și zecimal-binară, reprezintă o deosebită importanță datorită faptului că datele de calcul sînt reprezentate:

- în calculatoarele numerice în sistem binar
- pe suportii externi ai informațiilor, în sistem zecimal

Ca urmare, la introducerea numerelor în calculator, are loc o conversiune zecimal-binară, iar la extragerea rezultatelor din calculator are loc conversiunea binar-zecimală.

Conversiunea zecimal - binară a numerelor întregi

Se realizează pe baza regulilor de conversiune a numerelor întregi reprezentate în baza p în echivalentul lor în baza q .

Se aplică teorema unicității cîtului și restului

$$\text{Dacă : } N_p = a_n q^n + a_{n-1} q^{n-1} + \dots + a_1 q^1 + a_0 q^0$$

$$\text{Se observă că : } \frac{N_p}{q} = \text{cît } (C_0) + \frac{\text{Rest } (r_0)}{q}$$

unde C_0 = parte întreagă

r_0 = parte fracționară

Intrucît atît cîtul cît și restul sînt unice, egalînd părțile fracționare avem :

$$a_0 = r_0$$

$$C_0 = a_n q^{n-1} + \dots + a_1 q^0$$

Repetînd împărțirea cîtului C_0 prin q obținem

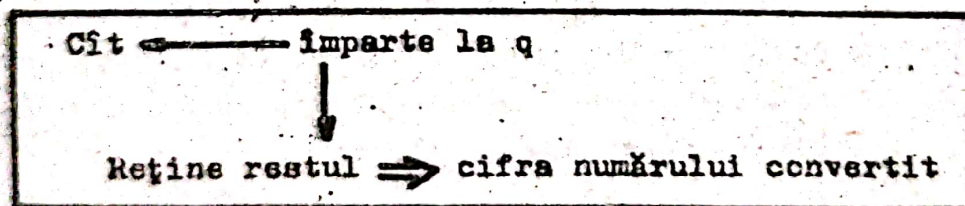
$$\frac{C_0}{q} = \text{cît } (C_1) + \text{Rest } (r_1), \text{ de unde}$$

$$a_1 = r_1 \quad | \quad C_1 = a_n q^{n-2} + \dots + a_2 q^0$$

Rezultă următoarea regulă de conversiune a numerelor întregi din baza p în baza q (metoda împărțirii repetate prin q)

- se împarte N_p la q , iar restul este a_0
- se împarte cîtul rezultat la q , iar restul este a_1
- se împarte cîtul rezultat la q , restul este a_2 , etc.

Această regulă se poate reprezenta prin următoarea schemă:



Aplicînd schema de mai sus pentru o conversiune zecimală-binară se obține:

$$\begin{array}{cccccccc}
 0 & \leftarrow & 1 & \leftarrow & 2 & \leftarrow & 5 & \leftarrow & 10 & \leftarrow & 21 & \leftarrow & 42 & \leftarrow & 85 & | & 2 & (85)_{10} = (1010101)_2 \\
 & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & & \\
 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & &
 \end{array}$$

Conversiunea zecimal - binară a numerelor subunitare

Se realizează pe baza regulilor generale de conversiune a numerelor subunitare din baza p în baza q :

Dat fiind numărul N_p , $0 < N_p < 1$, reprezentat în baza p

să se determine coeficienții $a_1, a_2, \dots, a_m$, fiecare mai mic decât q astfel ca :

$$N_p = a_1 q^{-1} + a_2 q^{-2} + \dots + a_m q^{-m}$$

înmulțind membrii egalității cu q :

$$qN_p = \underbrace{a_1}_{\text{întreg}} + \underbrace{a_2 q^{-1} + \dots + a_m q^{-m+1}}_{\text{parte fracționară}}$$

$$qN_p = \underbrace{u_{-1}}_{\text{parte întregă}} + \underbrace{v_{-1}}_{\text{parte fracționară}}$$

Egalind părțile întregi și fracționare între ele, rezultă :

$$a_1 = u_{-1}$$

$$v_{-1} = a_2 q^{-1} + \dots + a_m q^{-m+1}$$

Repetăm înmulțirea cu q :

$$qv_{-1} = a_2 + a_3 q^{-1} + \dots + a_m q^{-m+2}$$

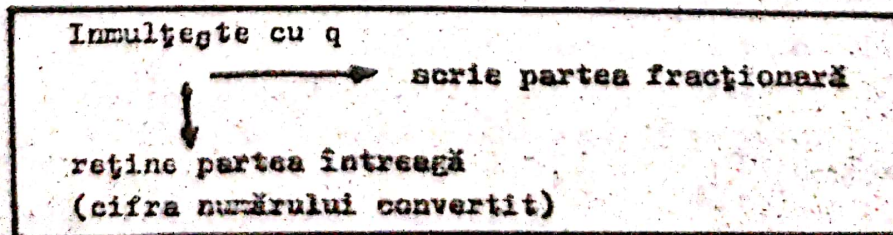
unde : $a_1 = a_2$ etc.

Rezultă următoarea regulă de conversiune a numerelor fracționare din baza p în baza q (regula înmulțirii repetate cu q)

- se înmulțește N_p cu q , iar partea întregă este a_1
- se înmulțește partea fracționară rezultată cu q , partea întregă este a_2
- se înmulțește partea fracționară rezultată cu q , partea întregă este a_3

Acest proces nu trebuie să aibă un sfârșit ca în cazul numerelor întregi; el se efectuează pînă la determinarea numărului dorit de cifre (cu cît numărul dorit de cifre determinate este mare, cu atît mai mare este precizia de reprezentare a numărului N_p în sistemul de numerație în baza q).

Procesul de conversiune se poate reprezenta prin schema de la stînga la dreapta :



Aplicând schema pentru numărul 0,4375 se obține:

$$\begin{array}{ccccccc} 0,4375 & \xrightarrow{\cdot 2} & 0,8750 & \xrightarrow{\cdot 2} & 1,7500 & \xrightarrow{\cdot 2} & 3,5000 \xrightarrow{\cdot 2} 0 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 0 & & 1 & & 1 & & 1 \end{array}$$

$$(0,4375)_{10} = (0,0111)_2$$

Obs. Conversiunea numerelor neîntregi mai mari ca 1

se realizează prin conversiunea separată a părții întregi și a părții fracționare prin metodele de mai sus. *urmata este concatenarea celor două*

Conversia binar - hexazecimală

Pentru conversia numerelor din sistem binar în sistem hexazecimal, se grupează câte 4 cifre binare (începând din partea dreaptă pentru numerele întregi, respectiv de la virgulă spre dreapta pentru numerele subunitare) și se scrie echivalentul lor în hexazecimal (pe baza tabelului 1.2).

Ex.: Se convertește numărul binar *intreg!*

$$\begin{array}{cccc} 0001 & 0111 & 1001 & 0011 \\ 1 & 7 & 9 & 1 \end{array}$$

în care grupând câte 4 cifre binare și scriind echivalentul lor în hexazecimal obținem: $N_{16} = 1791$

Conversia binar - zecimală

Metoda utilizată pentru conversiunea din sistemul binar în zecimal, se bazează pe procesul de scoatere în factori.

De exemplu se consideră un număr binar întreg

$$N_2 = a_4 \cdot a_3 \cdot a_2 \cdot a_1 \cdot a_0$$

Valoarea acestui număr calculată în zecimal este:

$$N_{10} = a_4 \cdot 2^5 + a_3 \cdot 2^4 + a_2 \cdot 2^3 + a_1 \cdot 2^2 + a_0 \cdot 2^0 \quad \text{sau}$$

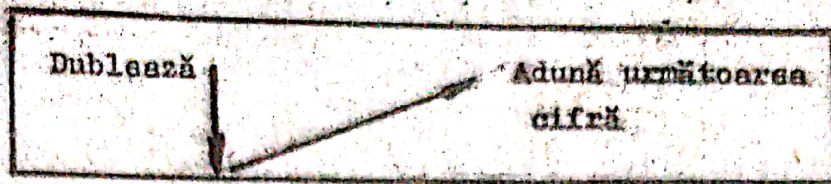
$$N_{10} = (((a_4 \cdot 2 + a_3) \cdot 2 + a_2) \cdot 2 + a_1) \cdot 2 + a_0$$

Coefficienții $a_0, a_1, \dots, a_4$ pot fi 0 sau 1 (deoarece s-a pornit de la reprezentarea binară a numărului).

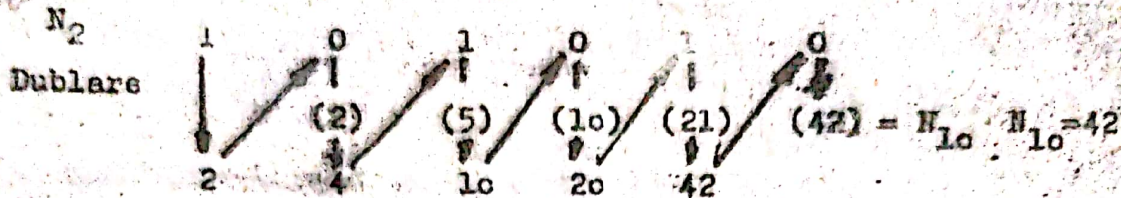
Rezultă următoarea regulă de conversiune a numerelor întregi binar în zecimal:

- se începe procesul de calcul cu coeficientul de rang cel mai mare a_n - se dublează a_n
- se adună a_{n-1} și se dublează rezultatul
- se adună a_{n-2} și se dublează rezultatul
-
- se adună a_1 și se dublează rezultatul
- se adună a_0

Acest proces de conversiune a unui număr întreg se poate reprezenta prin următoarea schemă:



Ex. : Să se convertească din binar în zecimal $N_2 = 101010$ aplicând schema indicată :



În mod asemănător se realizează conversia unui număr subunitar, dacă se consideră un număr binar subunitar :

$$N_2 = 0, a_{-1} a_{-2} a_{-3} a_{-4} a_{-5}$$

Valoarea acestui număr calculat în zecimal este :

$$N_{10} = a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + a_{-3} \cdot 2^{-3} + a_{-4} \cdot 2^{-4} + a_{-5} \cdot 2^{-5}$$

$$N_{10} = 2^{-1} (a_{-1} + 2^{-1} (a_{-2} + 2^{-1} (a_{-3} + 2^{-1} (a_{-4} + 2^{-1} \cdot a_{-5}))))$$

Rezultă următoarea regulă de conversiune a numerelor subunitare din sistemul binar în sistemul zecimal :

- se împarte a_{-n} cu 2 și se adună a_{-n+1}

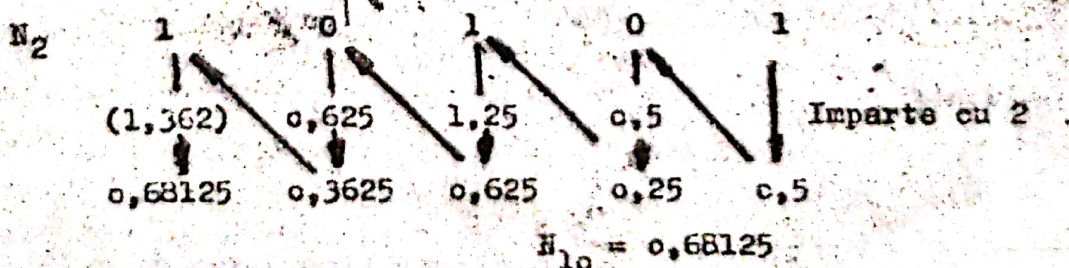
- se împarte rezultatul cu 2 și se adună a_{-1} ;

- se împarte rezultatul cu 2.

Procesul de conversie are loc după schema :



Exemplu: Să se convertească în zecimal numărul subunitar binar : $N_2 = 0,10101$



1.4. Structura generală și funcționarea unui calculator

1.4.1. Generalități

Calculatorul electronic este un sistem foarte complex de prelucrare automată a datelor. În descrierea unui calculator de capacitate medie sau mare trebuie avut în vedere că acesta este un ansamblu de două componente în interacțiune și anume:

- partea fizică (echipamentul) identificată prin termenul englez "hardware";
- partea logică (programele) identificată prin termenul englez "software".

Software-ul are două componente și anume:

- software-ul de bază sau sistemul de operare, care este o colecție de programe ce permite exploatarea echipamentului și
- software-ul de aplicații, care este o colecție de programe ce permite rezolvarea unor probleme de diverse tipuri.

Exploatarea calculatorului nu impune cunoașterea în amănunt a modului în care este construit sau funcționează acesta. Pentru înțelegerea corectă a unor aspecte privind modul de exploatare sunt necesare un minim de cunoștințe privitoare la construcția și funcționarea calculatorului.

1.4.2. Schema bloc a unui calculator

Calculatorului i se transmit informațiile de prelucrat constituite din date și comenzi. Calculatorul prelucrează datele pe baza comenzilor primite și furnizează utilizatorului rezultatele obținute. Astfel că o primă reprezentare schematică a unui calculator este dată în fig.1.2.

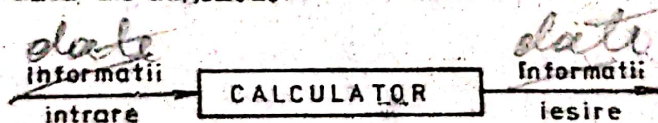


Fig.1.2. Schema bloc a unui calculator

Acest mod simplificat de prezentare a unui calculator ca o "cutie neagră" la care se știe ce se introduce și ce se extrage, dar nu se știe ce se petrece în interior nu poate ajuta la înțelegerea construcției și funcționării calculatorului. De aceea se detaliază în schema bloc din fig.1.3. părțile componente ale unui calculator. De subliniat că este vorba de o reprezentare

logică și nu fizică, reprezentare care pune în evidență funcțiile diferitelor părți componente numite unități.

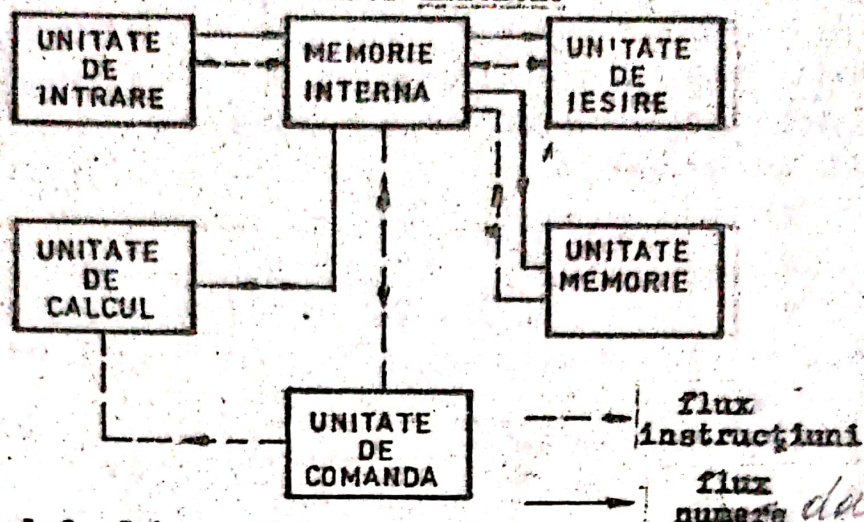


Fig. 1.3. Schema bloc a calculatorului electronic

Unitățile de intrare/ieșire - asigură transferul informațiilor (date și instrucțiuni) înspre și dinspre calculator. Cel mai frecvent se utilizează: cititorul de cartele, imprimanta, banda magnetică, perforatorul de cartele, display-ul etc. *monitor etc.*

Unitatea de calcul - execută operații aritmetice și operații de comparare a datelor. *logică asupra datelor*

Operațiile se execută asupra numerelor reprezentate în baza doi, operația aritmetică de bază fiind operația de adunare. Celelalte operații aritmetice se reduc prin diferite artificii la operația de adunare.

Unitatea de comandă - interpretează instrucțiunile introduse în memoria calculatorului și comandă celorlalte unități ale calculatorului executarea acestor instrucțiuni.

Unitatea de memorie. Memoria este acea parte a calculatorului care permite înregistrarea, stocarea și consultarea datelor. La memorie are acces unitatea de comandă și unitatea de calcul precum și unitățile de intrare/ieșire. Ea trebuie să satisfacă două cerințe contradictorii și anume: capacitate mare de stocare și timp de acces redus.

Din acest punct de vedere calculatorul are două tipuri de memorie și anume:

- memorie operativă sau centrală caracterizată de capacitatea de stocare redusă și timp de acces mic;
- o memorie externă cu capacitate de stocare mare și timp de acces ridicat.

Unitatea de control, unitatea de calcul și memoria operativă

constituie unitatea centrală (linia punctată din fig. 1.4).

Unități de schimburi multiple

Deoarece între viteza de lucru a unității de comandă și cea a unităților periferice există o mare deosebire se impune ca operațiile de transfer spre memorie și invers să se facă concomitent și independent de funcționarea unității de comandă.

Acest lucru este realizat de unitățile de schimburi multiple (fig. 1.4), care sînt de fapt calculatoare electronice specializate în transferul de date. Ele fac legătura cu unitățile periferice prin intermediul unor organe tampon numite unități de legătură.

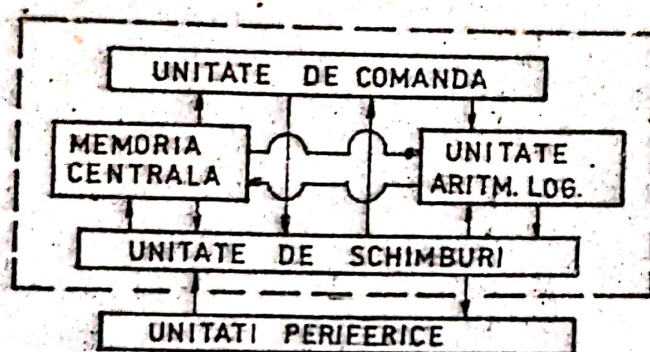


Fig. 1.4. Structura generală a unui calculator.

1.4.3. Descrierea unităților unui calculator

Această descriere nu este exhaustivă, ea se referă numai la amănuntele strict necesare înțelegerii modului de funcționare și utilizare a unui calculator.

Memoria internă

Elementul de bază al memoriei interne este inelul de ferită (fig. 1.5 a și b).

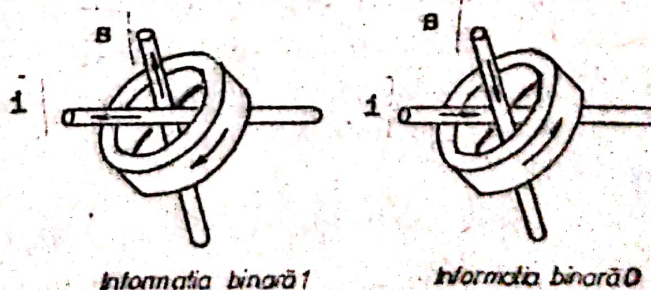


Fig. 1.5 Inel de ferită

Aceste inele sînt caracterizate de capacitatea de magnetizare sub acțiunea curentului electric (în două sensuri) și de o

curbă de histerezis magnetică constantă.

Celor două sensuri ale cîmpului magnetic li se pun în corespondență cifrele 0 și 1.

Schimbarea sensului de magnetizare se poate face cu ajutorul circuitelor electrice 1 și 5.

Inelele de ferită se montează în rețele de sîrmă dreptunghiulare de dimensiune $a \times b$ (fig.1.6).

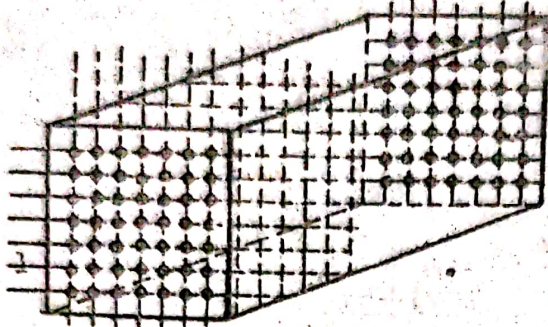


Fig.1.6 Rețea de inele

Timpul de acces - timpul necesar unei operații de scriere/citire - este de 940 ns ($1 \text{ ns} = 10^{-9} \text{ sec}$) iar debitul este de cca 2000000 octeți/sec.

Reprezentarea internă a informațiilor

Fiecare inel poate înmagazina cifra 0 sau 1.

Bit = cea mai mică unitate de informație memorizabilă.

(bit = binary digit = cifra binară).

Octet (byte) = cea mai mică unitate de memorie adresabilă și este format din 8 biți. Poate să îndeplinească funcțiile de scriere, păstrare sau citire a unei informații și ca atare poate fi adresată în memorie.

Deoarece, de regulă, informațiile nu se pot stoca pe un octet, se folosesc pentru reprezentarea acestora, multipli ai octetului și anume:

semicuvînt - un șir de doi octeți;

cuvînt - un șir de patru octeți;

dublu cuvînt - un șir de 8 octeți

În calculatorul FELIX locația de bază este cuvîntul organizat pe 32 biți deci 4 octeți.

Rețele dreptunghiulare (v. fig.1.6) se organizează în module, fiecare modul avînd 2^{14} octeți.

Un bloc are 1 - 4 module, iar o memorie internă are 1 - 4 blocuri de memorie.

Un Kilo octet (Ko) are 1024 octeți = 2^{10} octeți.

Un modul are 16 Ko, un bloc 16 - 64 Ko, iar unitate de me-

memorie a calculatorului FBILIX C-256 are cel mult 256 Ko.

Pentru a putea manipula informațiile în memorie, este necesar ca locațiile în care se depozitează acestea să fie găsite. Acest lucru se realizează atribuind fiecărei locații o adresă.

Adresa este un număr întreg format din 18 cifre exprimate în binar sau 5 cifre în hexazecimal.

Memoria externă are această denumire pentru că nu face parte din unitatea centrală.

Cele mai utilizate echipamente ca memorie externă sînt unitățile de bandă magnetică și unitățile de discuri magnetice.

Banda magnetică

Inregistrarea pe bandă se face pe 8 p i s t e, o bandă avînd o lățime de 12,5 mm și o lungime de 730-750 m.

Densitatea de înregistrare: 31-62 biți/mm;

Capacitatea de depozitare: funcție de lungimea benzii,

Viteza de defilare: 190 cm/sec.

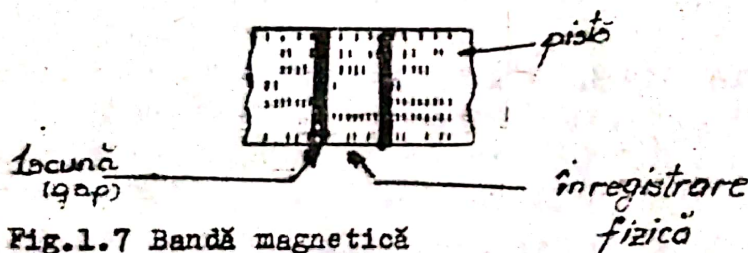


Fig.1.7 Bandă magnetică

Discul magnetic

Inregistrarea pe discuri se face pe cercuri concentrice numite piste împărțite în sectoare (fig. 1.8).

Informațiile se înregistrează pe 10 din cele 12 fețe (discurile DIAM), sau pe cele 20 din cele 22 de fețe (discurile DIMAS) ale pachetului de 6, respectiv 11 discuri.

Fiecare față conținînd 203 piste, un pachet conținînd 203 cilindri.

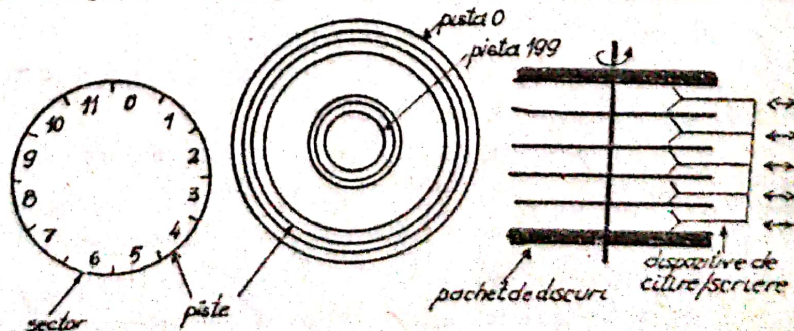


Fig.1.8. Disc magnetic.

Capacitatea unui pachet DIAM: 6144 Ko octeți, iar a unui pachet DIMAS: 24576 Ko.

Debit: 156 Ko/s și respectiv 312,5 Ko/s.

Citirea / scrierea informațiilor este realizată de la capete de citire/scriere care au acces radial în timp ce discurile se rotesc cu o viteză de 2400 rot/min.

Unități de intrare/ieșire

Cele mai utilizate sînt, mai ales de începători, cititorul de cartele și imprimanta.

Cititorul de cartele asigură citirea unui număr de pînă la 1200 cartele/min. Cartela este un suport standardizat de dimensiune 18,7 x 6,25 cm organizată pe 12 linii și 80 coloane.

Fiecare caracter FORTRAN se înregistrează pe cartelă ca o succesiune de 1 pînă la 3 perforații (v. fig. 1.9).

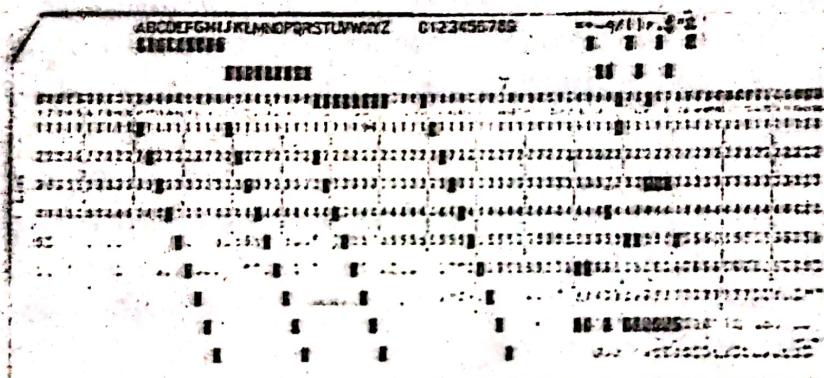


Fig. 1.9. Cartelă perforată.

Imprimanta este un dispozitiv cu ajutorul căruia se tipăresc datele ce interesează utilizatorul. Tipărirea se face pe hîrtie avînd pe rînd 80 sau 132 caractere, cu o viteză de pînă la 1200 rînduri/min. Imprimarea se face prin diferite metode mecanice, prin intermediul unor organe tampon numite unități de legături.

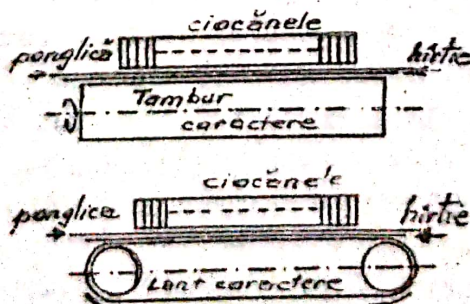


Fig. 1.10. Imprimanta

1.4.4. Principiile fundamentale de lucru ale calculatorului

Calculatorul numeric efectuează în principal operații aritmetice cu numere și în particular adunarea, scăderea, înmulțirea, împărțirea. Așa cum s-a arătat mai sus aceste operații se reduc de fapt la operația de adunare. Pentru a înțelege însă modul cum funcționează un calculator trebuie să se rețină principiile fundamentale de lucru ale calculatorului.

I. Memoria calculatorului este conservativă și substitutivă.

Acest principiu este ușor de înțeles comparând memoria calculatorului cu banda magnetică a unui casetofon.

O melodie înregistrată se conservă, ea putând fi ascultată teoretic de un număr nelimitat de ori.

Dacă se imprimă o nouă melodie, în mod automat melodia imprimată anterior se șterge, deci noua melodie se substituie celei vechi. În mod asemănător se petrec lucrurile și cu memoria calculatorului.

Dacă extragem conținutul unei adrese, acesta se conservă mai departe la respectiva adresă. Pentru a șterge acest conținut există două posibilități:

1. introducerea unei noi informații la această adresă, care se substituie celei vechi, sau
2. se transmite o instrucțiune de ștergere a conținutului respectivei adrese.

Această ultimă operație se face respectând cel de-al doilea principiu.

II. Instrucțiunile lucrează cu adrese și nu cu conținutul lor

Acest mod de lucru este analog cu modul de lucru folosit în algebră unde se operează cu simbolurile x , y , z și nu cu numere.

De exemplu, trebuie efectuată înmulțirea numărului 7 cu numărul 1,7; această operație se execută astfel:

Se memorează la o adresă oarecare, de exemplu 022, numărul 7, iar la o adresă 028, numărul 1,7. Instrucțiunea de executare a operației dorite va comanda înmulțirea conținutului adresei 022 cu conținutul adresei 028, sau, succint, în limbajul programatorilor "înmulțește pe 022 cu 028" și nu înmulțirea numărului 7 cu numărul 1,7.

Prin acest mod de a scrie instrucțiunile, programul odată întocmit rămâne valabil pentru orice valori ale numerelor păs-trate în celulele cu adresele de mai sus, cunoscându-se că

conținutul acestora poate fi schimbat fără a afecta marșul general al calculului.

Deoarece atât instrucțiunile cât și numerele sînt depozitate sub formă cifrică trebuie enunțat un alt principiu care spune:

III. Conținutul fiecărei adrese poate fi interpretat fie drept număr, fie drept instrucțiune

Pentru a nu se produce confuzii calculatorului trebuie să se indice de la început în care adrese se găsesc ^{date} numere și în care adrese se găsesc instrucțiuni (comenzi). În caz contrar conținutul adresei respective va fi tratat ca unitatea de comandă drept instrucțiune iar de către dispozitivul aritmetic drept număr. Acest al treilea principiu are implicații deosebit de importante, deoarece calculatoarele pot avea în unele situații comportări asemănătoare ființelor raționale, de unde și apelativul de "creier electronic" folosit uneori.

1.4.5. Modul de lucru al calculatorului numeric

În mod obișnuit operațiile aritmetice folosesc două numere denumite operanzi sau argumente, formînd în urma operațiilor un al treilea număr numit rezultat.

Calculatorul este informat asupra operațiilor pe care trebuie să le efectueze prin intermediul instrucțiunilor grupate într-un program. Într-o instrucțiune se specifică atât operația care urmează a fi efectuată cât și operanzii care participă în operația respectivă. De exemplu, se cere efectuarea calculului indicate de expresia:

$$y = \frac{x}{(a + b)d} - c$$

Dacă valorile variabilelor a, b și c au fost introduse în memoria calculatorului, succesiunea instrucțiunilor care conduce la rezolvarea problemei este dată de tabelul 1.3.

Tabelul 1.3.

Instrucțiunea	Operația	Primul operand	Al doilea operand	Rezultat
Instrucțiunea 1	adunare	a	b	z
Instrucțiunea 2	înmulțire	z	d	t
Instrucțiunea 3	împărțire	x	t	v
Instrucțiunea 4	scădere	v	c	y

Operațiile aritmetice indicate de expresia din exemplu se execută în unitatea de calcul aritmetic și logic (v.fig.1.3). Această unitate este constituită din mai multe dispozitive, numite registre, dintre care cel mai important se numește

Acumulator . In acest registru se vor efectua calculele astfel:

- în cazul instrucțiunii 1 din tabelul 1.3
- a) se aduce în acumulator conținutul adresei la care e memorat operandul a;
- b) se înmulțește conținutul adresei la care e memorat operandul b cu conținutul Acumulatorului, rezultatul obținut fiind depozitat de asemenea în Acumulator (z);
- c) rezultatul z din Acumulator este transferat în memorie la o adresă oarecare, Acumulatorul rămânând disponibil pentru o nouă operație. Se observă din cele de mai sus respectarea riguroasă a principiilor I și II enunțate anterior.

1.5. Reprezentarea informațiilor în calculator

La intrarea în calculator informațiile se prezintă sub formă de șiruri de caractere FORTRAN care urmează a fi reprezentate în interesul calculatorului în formă binară.

Din acest punct de vedere în calculator se introduc :

- informații alfanumerice
- informații numerice

1.5.1. Informația alfanumerică - Informația alfanumerică este reprezentată în calculator prin intermediul unor grupuri de 4 biți de fiecare caracter pe baza codului EBCDIC (Extending Binary Coded Decimal Interchange Code), prezentat în tabelul nr.

1.3. Astfel șirul alfa numeric TEXTIL se reprezintă în calculator prin :

1110	0011	1100	0101	1110	0111	1110	0011	1100	1001	1101	0011
E	3	C	5	E	7	E	3	C	9	D	3
T		E		X		T		I		L	

Fieci fiecare caracter se reprezintă pe un grup de patru biți.

1.5.2. Informația numerică

Informația numerică este formată din :

- numere întregi
- numere reale

Reprezentarea acestor numere se face în binar pentru numerele întregi, în așa numita reprezentare în "virgulă fixă".

Reprezentarea în virgulă fixă se face pe un cuvânt, deci pe

32 biți, care au următoarea structură :

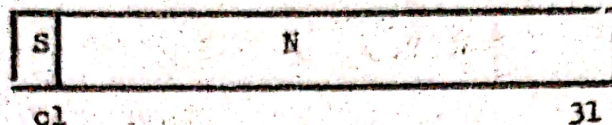


Fig.1.11 : Reprezentarea numerelor întregi

Bitul 0 este bit de semn și prin convenție s-a stabilit ; bitul zero - semn pozitiv; bit 1 - semn negativ. Valorii numărului N îi sunt rezervate 31 biți deci într-un cuvânt se poate reprezenta o plajă a numerelor întregi cuprinse între :

$$- 2^{31} \leq N \leq 2^{31} - 1$$

aceasta însemnând :

$$- 2147483648 \leq N \leq 2147483647$$

Reprezentarea în virgulă mobilă

Se folosește pentru reprezentarea sub forma cvasilogaritmă de scriere a numerelor reale

$$\begin{matrix} + \\ - \end{matrix} M \cdot B^{\begin{matrix} + \\ - \end{matrix} E}$$

în care M este mantisa numărului normalizată (prima cifră de după virgulă diferită de zero) ; B este baza de numerație, iar E este puterea la care se ridică baza,

Mantisa trebuie să aibă valoarea :

$$\frac{1}{16} \leq M < 1$$

În sistemul de calcul FELIX reprezentarea numerelor reale se face în două moduri și anume reprezentarea în simplă precizie pe un cuvânt (32 biți) și în dublă precizie pe un dublucuvânt (64 biți).

La reprezentarea în simplă precizie repartitia în structura unui cuvint este :

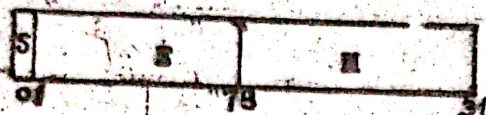


Fig.1.12 Reprezentarea în simplă precizie

- adică :
- bitul zero este bit de semn
 - următorii 7 biți sînt rezervați exponentului E corectat cu valoarea 64 numită caracteristica C

$$0 \leq C \leq 128 \text{ deoarece } C = E + 64$$

~~Exponentul~~ exponentului se face pentru evitarea semnului negativ, deci folosirea unui bit pentru semnul minus. Se folosește corectarea automată a lui E.

- ultimii 24 biți sînt rezervați pentru reprezentarea mantisei M.

La reprezentarea în dublă precizie crește numărul de biți alcași reprezentării mantisei M de la 24 la 55 așa încît în simplă precizie numărul de cifre semnificative ale mantisei este de 6 cifre zecimale iar la dubla precizie este de 15 cifre zecimale. Plaja de reprezentare a numerelor reale în calculator este de

$$5,4 \times 10^{-79} \leq M \leq 7,2 \times 10^{75}$$

CAPITOLUL 2

ALGORITMI SI SCHEME LOGICE

2.1. Etapele preliminare rezolvării unei probleme pe calculatorul electronic

Pentru rezolvarea unei probleme cu ajutorul calculatorului electronic este necesară parcurgerea mai multor etape preliminare și anume:

1. problema trebuie să fie pusă sub formă matematică, iar datele să fie puse sub formă numerică;

2. precizarea datelor de intrare/iesire;

3. a alegerea metodei sau a metodelor numerice de rezolvare.

Este știut că pentru rezolvarea aceluiași probleme se pot folosi mai multe metode, diferența dintre ele constând în durata procesului de rezolvare, mărimea erorilor. Descrierea acestor metode se numește algoritm. Un algoritm este caracterizat prin trei proprietăți, și anume: claritate, universalitate, finitudinea.

Prin claritate se înțelege că propozițiile enunțate sunt exprimate precis, corect, fără ambiguități.

Universalitatea sau generalitatea arată că algoritmul se aplică unei clase de probleme de același tip, iar finitudinea arată că într-un timp determinat, după un anumit număr de pași se obține un rezultat.

4. Intocmirea schemei logice care descrie grafic algoritmul ales.

Cum unele probleme se pot rezolva prin mai multe metode, vor exista tot atâtea variante de scheme logice.

5. Intocmirea programului într-un limbaj înțeles de calculator.

Interpretarea și descrierea activității blocurilor

Blocul de intrare/ieșire, de forma unui paralelogram, indică efectuarea unei operații de intrare/ieșire, respectiv citirea valorilor unor variabile sau afișarea unor rezultate ale activității programului sau a unor valori citite anterior. Pentru a pune în evidență care este funcțiunea lui particulară în schemă, în bloc se va înscrise în mod obligatoriu cuvântul CITEȘTE sau SCRIE după caz.

Blocul de calcul, a cărui reprezentare grafică este un dreptunghi, indică efectuarea unor operații aritmetice indicate de expresiile înscrise în bloc. Ca urmare a activității blocului, variabilei aflată la stînga semnului " = " i se atribuie valoarea rezultată din membrul drept.

Blocul de inițializare, reprezentat printr-un dreptunghi (caz particular al blocului de calcul). Se folosește pentru atribuirea de valori inițiale unor variabile în cazul unui proces iterativ de calcul.

Blocul de avans, reprezentat printr-un dreptunghi (caz particular al blocului de calcul), indică operația de calcul la care este supusă o variabilă, denumită contor sau variabilă de ciclare.

Blocul procedură sau subprogram - reprezentat de un hexagon - indică un grup de operații care se efectuează în cadrul unui subprogram.

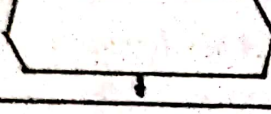
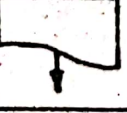
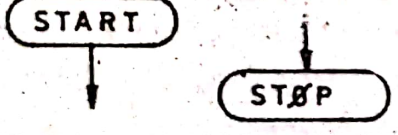
Conectorii pe pagină asigură obținerea unor scheme logice clare, ușor de urmărit, deoarece în locul unei linii se folosesc doi conectori care se desenează în punctul de plecare ($\rightarrow \textcircled{1}$) și în punctul de sosire ($\textcircled{1} \rightarrow$) al liniei pe care o înlocuiesc.

Pentru a păstra corespondența univocă între cei doi conectori, în interiorul cercurilor se înscrise aceeași cifră arabă, de regulă.

Conectorii de pagină asigură continuitatea schemelor ce se desfășoară pe mai mult de o pagină. Conectorul ce termină o pagină și cel ce începe pagina următoare vor avea înscrise același caracter alfabetic majuscul.

Blocul delimitator de schemă, reprezentat prin dreptunghiul curbiliniu, indică limitele unei scheme. În interiorul acestuia se scrie cuvântul START/STOP, PORȚIT/OPRIT, după caz.

Blocul logic sau de decizie, reprezentat printr-un romb, indică luarea unei decizii pe baza analizei unei condiții.

SIMBOL	DENUMIRE
	Blocul operațiilor de intrare/ieșire
	Blocul de decizie sau blocul logic
	Blocul operațiilor de calcul (de inițializare, de avans)
	Blocul procedură (subprogram)
	Conectori pe pagină
	Conectori de pagină
	Blocul imprimantei
	Blocuri delimitatoare de schema

Există trei tipuri de scheme logice corespunzător celor trei tipuri de algoritmi, și anume :

- scheme logice liniare
- scheme logice cu ramificații
- scheme logice cu cicluri
 - cu număr cunoscut de pași
 - cu număr necunoscut de pași

2.2. Schema logice de calcul

Programul, în baza căruia calculatorul rezolvă o problemă, se întocmește numai dacă s-au stabilit în mod clar, pas cu pas etapele de rezolvare a acesteia. În forma narativă, algoritmul este de cele mai multe ori greu de urmărit și aceasta cu cât problema este mai complicată. Din acest motiv s-a trecut la reprezentarea grafică a acesteia.

Folosind reprezentarea prin schema logică a algoritmului se poate urmări cu multă ușurință etapele acesteia, este posibil schimbul de informații sau se poate secționa problema în vederea rezolvării ei de către mai mulți programatori.

Pentru desemnarea aceleiași noțiuni, definite mai sus, în literatura de specialitate se mai întâlnesc noțiunile de organigramă sau schemă de calcul sau diagramă de calcul.

În cursul de față simbolurile folosite la întocmirea schemelor logice sînt prezentate în tabelul nr.2.1.

În interiorul schemelor logice se folosește fie scrierea obișnuită, fie cu caractere FORTRAN. În curs s-a optat pentru a doua metodă, pentru a familiariza cititorul cu acest mod de scriere. Trebuie subliniat că în schemele logice cît și pe parcurs, pentru a nu complica modul de scriere, s-a utilizat semnul "=" nu cu sensul de egalitate ci de atribuire.

Reguli pentru întocmirea schemei logice

1. Schema logică se întocmește de sus în jos.
2. Începutul și sfîrșitul unei scheme logice sînt marcate de blocurile delimitatoare corespunzătoare.
3. Legătura între blocurile schemei logice se face prin săgeți.
4. Pentru evitarea intersecțiilor săgeților, în vederea obținerii unor scheme clare, se folosesc conectorii pe pagină.
5. Dacă pentru o schemă e necesar mai mult de o pagină, legătura între părțile schemei de pe diferite pagini se face prin conectori de pagină.
6. Orice deplasare laterală trebuie urmată obligatoriu de revenire la verticala schemei.

2.2.1. Scheme logice de calcul liniare

Aceste scheme se numesc liniare pentru că desfășurarea procesului de calcul se desfășoară liniar de sus în jos.

Exemplu : Să se întocmească schema logică pentru calculul relației 2.1.

Obs. Asupra modului de transcriere folosit în schemă vezi cap.3.

$$Y = \frac{x}{a+b} \cdot d - c$$

(2.1)

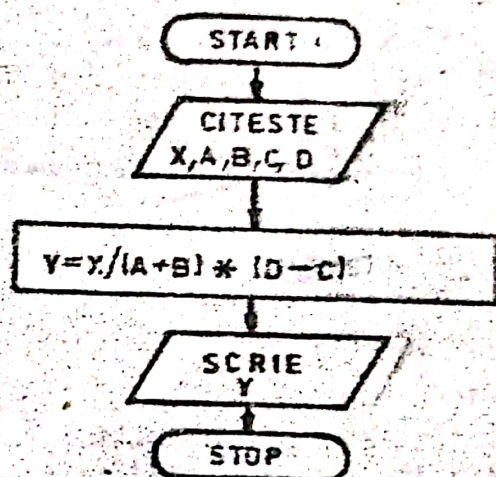


Fig.2.1. Schemă logică liniară

Din schemă se observă că, imediat după blocul START, s-a folosit blocul de intrare care indică citirea valorilor variabilelor x, a, b, d, c date prin problemă.

Ca urmare a acestei operații, în memoria calculatorului se vor depozita valorile variabilelor respective.

După operația de citire, apare blocul de calcul, care indică operațiile ce trebuiesc efectuate cu valorile citite.

În urma calculului rezultă o valoare ce se atribuie variabilei Y , valoare care se va depozita și ea în memoria calculatorului. Pentru a afla valoarea aceasta se cere afișarea printr-un bloc de ieșire.

Operația fiind terminată schema se termină prin blocul delimitator STOP.

2.2.2. Scheme logice de calcul cu ramificații

Exemplu: Să se întocmească schema logică de calcul a expresiei :

$$y = (a + b) / \sqrt{a^2 - c} \quad (2.2)$$

Se observă că expresia are sens dacă expresia de sub radical este pozitivă sau egală cu zero, ca atare schema logică va avea forma din fig.2.2.

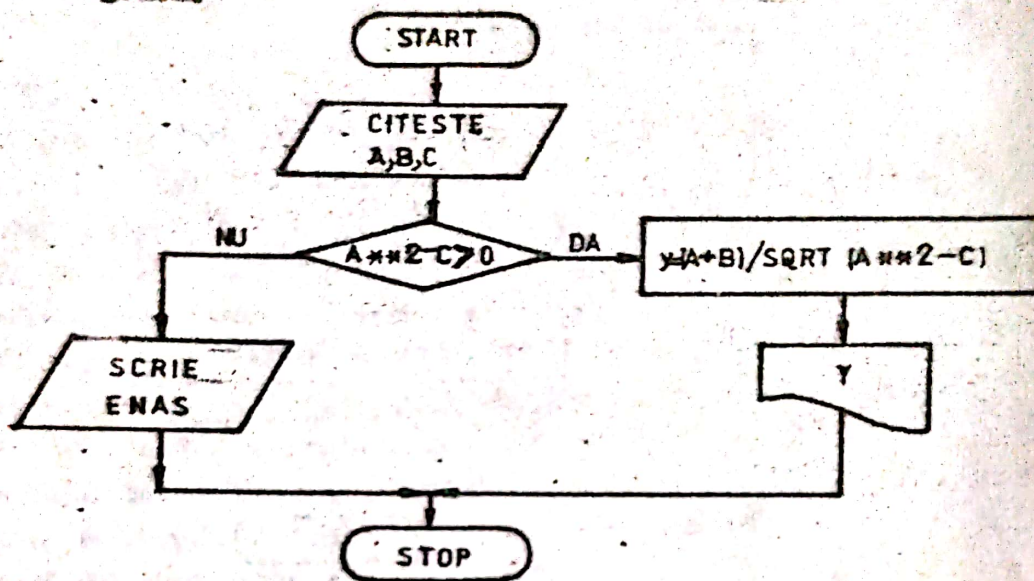


Fig. 2.2 Schema logică cu ramificații

După citirea valorilor variabilelor A, B și C se folosește blocul de decizie prin care se compară expresia înscrisă cu zero. În urma comparării se poate obține un rezultat pozitiv (ramura DA), sau un rezultat negativ (ramura NU).

Pentru cele două situații există două moduri de rezolvare, de unde necesitatea ramificației.

Dacă $A^2 - C \leq 0$ se cere calculatorului afișarea la imprimantă a unui mesaj care să indice acest lucru.

De exemplu, în cazul exercițiului propus ENAS sînt inițialele cuvintelor "Expresia nu are sens". După afișare se revine în verticala principală a schemei la blocul STOP. Pe ramura pozitivă operațiile se desfășoară ca și în cazul schemei liniare.

Deci la schemele cu ramificații este important ca orice ramificație să se încheie prin revenirea în verticala principală a schemei în blocul, corespunzător cu logica problemei.

2.2.3. Scheme logice cu cicluri

Aceste scheme se folosesc atunci cînd prin problema propusă apare necesitatea repetării unui grup de operații. Este de fapt situația care justifică în cel mai înalt grad utilizarea calculatorului, care trebuie folosit în probleme cu caracter repetitiv.

Aceste cicluri pot fi cu:

- a) număr cunoscut de pași
- b) număr necunoscut de pași

2.2.3.1. Scheme cu cicluri cu număr cunoscut de pași

Exemplu: Să se întocmească schema logică pentru calculul mediei aritmetice a termenilor unui șir X cu N valori. Formula de calcul clasică este:

$$Y = \frac{1}{N} \sum_{i=1}^N X_i \quad (2.3)$$

Prezentată sub această formă, formula sumei nu permite a distinge ușor care operațiune trebuie repetată în mod identic de mai multe ori, însă de fiecare dată cu alte numere.

De aceea se va scrie relația de mai sus sub forma:

$$Y = \frac{1}{N} (S + X_1) \quad (2.4)$$

unde

$$S = \sum_{k=1}^{i-1} X_k \quad (2.5)$$

Acum lucrurile sînt clare. Într-adevăr operațiunea $s + X_1$ trebuie repetată de N ori. Să observăm acum modul de trecere de la etapa "i" la etapa următoare "i+1". Aceasta va efectua aceeași operațiune ca mai înainte, însă cu alte numere

$$s + X_{i+1} \quad (2.6)$$

unde s este rezultatul etapei precedente, adică

$$s = s + X_i \quad (2.7)$$

Această formulă însă are un aspect ciudat, deoarece din punct de vedere algebric ar apare că:

$$X_i = s - s = 0 \quad (2.8)$$

Ori, acest lucru nu este adevărat întotdeauna, valorile X_i sînt de regulă diferite de zero.

Pentru a interpreta corect relația 2.8 să ne reamintim că semnul „+” în cursul de față înseamnă atribuire.

Astfel, expresia de mai înainte arată de fapt că: la valoarea lui s se adaugă o nouă valoare x_1 , iar rezultatul sumării se atribuie variabilei s . Dacă ținem cont că această nouă valoare a lui s se va memora în locația de memorie atribuită mărimii s și că această memorare se face prin ștergerea valorii precedente

(principiul I al funcționării calculatorului), vom utiliza pentru mărimea s din membrul drept apelativul de valoare trecută sau precedentă deoarece ea există în momentul începerii operației de calcul, iar pentru mărimea s din membrul stâng apelativul de valoare actuală sau prezentă deoarece ea apare în momentul efectuării calculului și înlocuiește vechea valoare. În aceste noi coordonate verbale expresia 2.8 se interpretează în felul următor:

„La valoarea precedentă a variabilei s se adaugă valoarea variabilei x_1 , rezultatul obținut se atribuie variabilei s ”.

În aceste condiții expresia reprezintă de fapt operația care va trebui repetată de n ori.

Trecerea la etapa „ $i + 1$ ” a ciclului inductiv se face prin avansul cu un pas atribuind lui „ i ” o valoare „ $i + 1$ ” sau în notația folosită,

$$i = i + 1 \quad (2.9)$$

Această comandă modifică instrucțiunea ce realizează operația din expresia 2.7. Este o ilustrare a celui de-al treilea principiu de funcționare a calculatoarelor. Terminarea ciclului trebuie să aibă loc atunci când $i > N$. Dacă această condiție este satisfăcută, atunci se execută un transfer condiționat de ieșire din ciclu și se execută ultima operație

$$x = s/N \quad (2.10)$$

Pentru a sfârși cu începutul să vedem cum se execută operațiunea de bază în primul ciclu. Deci pentru $i = 1$ obținem $s = s + x_1$ (2.7) unde s , la care se adună x_1 , reprezintă conținutul adresei din memorie unde se depozitează valoarea variabilei s (principiul II al funcționării calculatoarelor). Deoarece la adresa respectivă poate exista de la executarea anterioară a altui program, un număr oarecare (conform principiului conservativității memoriei), trebuie avut grijă ca înainte de începerea ciclului să atribuim variabilei s valoarea zero:

Conversiunea zecimal - binară a numerelor întregi

Se realizează pe baza regulilor de conversiune a numerelor întregi reprezentate în baza p în echivalentul lor în baza q .

Se aplică teorema unicității cîtului și restului

$$\text{Dacă : } N_p = a_n q^n + a_{n-1} q^{n-1} + \dots + a_1 q^1 + a_0 q^0$$

$$\text{Se observă că : } \frac{N_p}{q} = \text{cît } (C_0) + \frac{\text{Rest } (r_0)}{q}$$

unde C_0 = parte întreagă

r_0 = parte fracționară

Intrucît atît cîtul cît și restul sînt unice, egalînd părțile fracționare avem :

$$a_0 = r_0$$

$$C_0 = a_n q^{n-1} + \dots + a_1 q^0$$

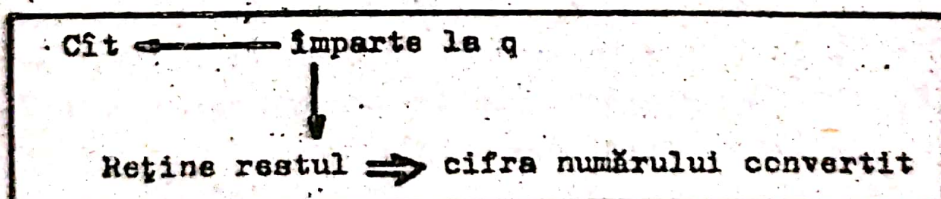
Repetînd împărțirea cîtului C_0 prin q obținem

$$\frac{C_0}{q} = \text{cît } (C_1) + \frac{\text{Rest } (r_1)}{q}, \text{ de unde}$$

$$a_1 = r_1 \quad \left| \quad C_1 = a_n q^{n-2} + \dots + a_2 q^0\right.$$

Rezultă următoarea regulă de conversiune a numerelor întregi din baza p în baza q (metoda împărțirii repetate prin q)

- se împarte N_p la q , iar restul este a_0
 - se împarte cîtul rezultat la q , iar restul este a_1
 - se împarte cîtul rezultat la q , restul este a_2 , etc.
- Această regulă se poate reprezenta prin următoarea schemă:



Aplicînd schema de mai sus pentru o conversiune zecimală-binară se obține:

$$\begin{array}{cccccccc|c}
 0 & \leftarrow & 1 & \leftarrow & 2 & \leftarrow & 5 & \leftarrow & 10 & \leftarrow & 21 & \leftarrow & 42 & \leftarrow & 85 & | & 2 & (85)_{10} = (1010101)_2 \\
 & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & & \\
 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & &
 \end{array}$$

Conversiunea zecimal - binară a numerelor subunitare

Se realizează pe baza regulilor generale de conversiune a numerelor subunitare din baza p în baza q :

Dat fiind numărul N_p , $0 < N_p < 1$, reprezentat în baza p

$$s = 0$$

(2.11)

Același raționament se aplică și în cazul variabilei i , variabilă de ciclare. Atribuim lui i ,

$$i = 1$$

(2.12)

indicând astfel calculatorului de la ce rang trebuie începută operația.

În final, cele mai sus discutate nu fac decât să exprime cu alte cuvinte ceea ce face un operator, care, efectuând calculul aceleiași medii, dorește ca:

- suma termenilor să fie corectă, deci va efectua suma din aproape în aproape, luând în lucru primii doi termeni, la rezultatul căreia adaugă pe al treilea, etc;

- să execute suma tuturor termenilor începând cu cel mai din stînga;

- să considere suma efectuată cînd s-au sumat toți cei N termeni ai șirului. În aceste condiții schema logică are următoarea formă:

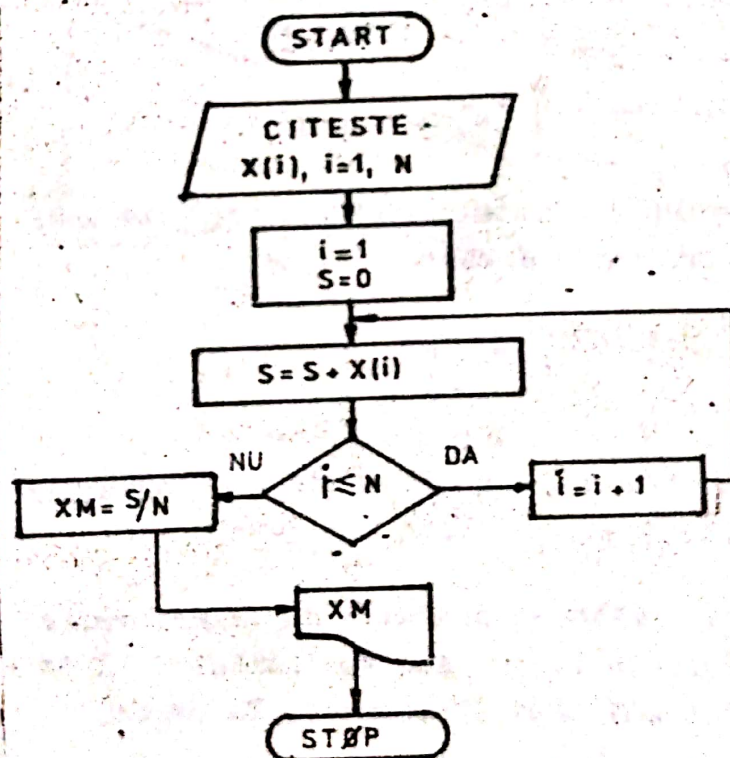


Fig. 2.3 Schemă logică cu cicluri cu număr cunoscut de pași

în care XM este $\bar{x}$.

2.2.3.2. Scheme logice cu cicluri cu număr necunoscut de pași

Aceste scheme logice se întocmesc atunci când procesul de calcul ciclic se desfășoară pe un număr necunoscut de pași. Condiția ieșirii din ciclu apare când s-a îndeplinit condiția impusă de problemă și nu după parcurgerea unui anumit număr de pași.

Exemplu:

Să se calculeze toate elementele unui șir mai mic decât o valoare M dată, cunoscându-se primii doi termeni ai șirului, 0 și 1, iar un termen din șir este suma precedentilor doi. Această șir se mai numește și șirul lui Fibonacci.

Pentru început să stabilim variabilele cu care se va lucra; este nevoie de trei variabile pentru a respecta regula de mai sus. Fie aceste trei variabile I , J , K astfel încât $K = I + J$. Aceasta este regula de bază.

Valorile inițiale date de problemă vor fi:

$$I = 0$$

$$J = 1$$

Scrind unele sub altele trei operații de calcul a tot atâția termeni se observă că:

$$K \quad J \quad I$$

$$1 = 1 + 0$$

$$1 = 1 + 1$$

$$3 = 1 + 2$$

În fiecare următoare operație de sumare variabila I are precedenta valoare a lui J ; iar variabilei J i se atribuie vechea valoare a lui K , de unde regula de avans.

$$I = J$$

$$J = K$$

Condiția ieșirii din ciclu va fi constituită de îndeplinirea condiției: noua valoare calculată a lui K să fie mai mare decât valoarea M cunoscută. În caz contrar operația de calcul continuă. Din cele de mai sus se observă că numărul de pași

pe care îl face procesul până la obținerea rezultatului final nu este cunoscut.

Dacă noua valoare calculată K aparține sirului, se impune afișarea acesteia, operație specificată în schemă de blocul imprimantei, așezat la ieșirea pe ramura DA a blocului logic.

Schema logică este prezentată în figura de mai jos:

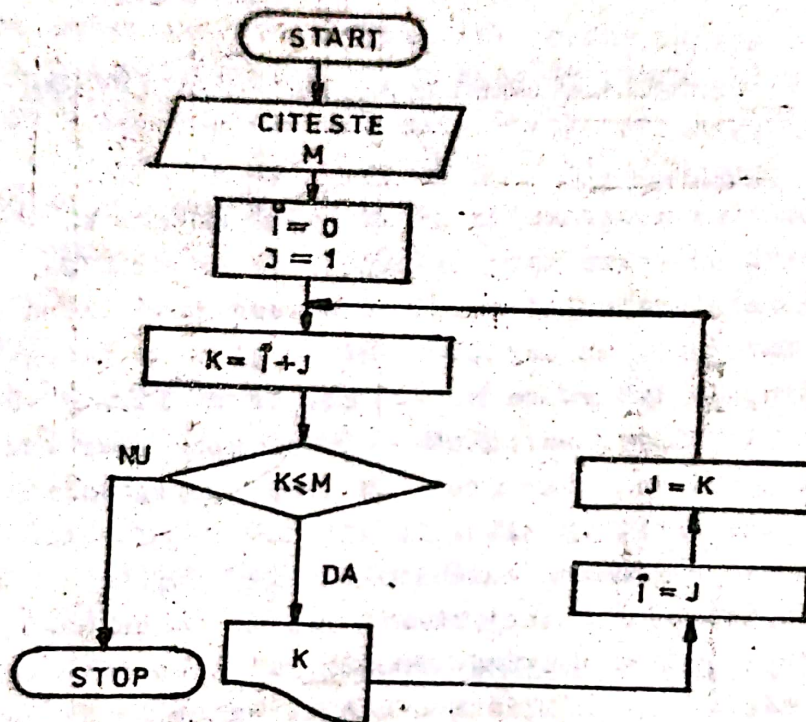


Fig. 2.4 Schemă logică cu cicluri cu număr necunoscut de pași.

CAPITOLUL 3

ELEMENTE DE BAZA ALE LIMBAJULUI FORTRAN

3.1. Generalități

Intecmirea programelor în limbajul intern al calculatoarelor necesită un volum mare de muncă. Pe de altă parte, programarea în limbajul calculatorului este specifică fiecărui tip de calculator, învățarea sa cerând un efort și timp îndelungat, iar

cunștințele dobândite nu mai pot fi utilizate odată cu trecerea la un alt tip de calculator. La acestea s-ar adăuga și posibilitatea apariției unor erori în timpul programării, erori mult mai frecvente decât în situația utilizării programării automate. Limbajele de programare automată nu sînt legate de particularitățile constructive ale unui anumit tip de calculator, ele reprezintă o formă de codificare a informației mai apropiată de utilizator.

Sceul urmărit prin utilizarea acestor limbaje îl constituie întecmirea programelor asemănătoare formei de exprimare a relațiilor din matematică. Datorită acestui avantaj crește eficacitatea muncii de programare, în timp ce probabilitatea apariției erorilor scade simțitor. Un limbaj de programare este specific unei anumite categorii de activități. Cele mai cunescute limbaje de programare automată, utilizate pentru rezolvarea problemelor cu caracter tehnic și științific sînt limbajele ALGOL și FORTRAN.

În cazul limbajului de programare-FORTRAN numele s-a obținut din contragerea a două cuvinte englezești: FORmula TRANslation, și sugerează totodată care este domeniul de aplicabilitate al acestuia - rezolvarea problemelor tehnice-științifice.

Prima variantă a fost elaborată în anii 1954-56 în cadrul corporației IBM (International Business Machines Corporation). Au apărut apoi variante îmbunătățite, cea prezentată în cursul de față fiind varianta FORTRAN IV.

Trebuie subliniat că acest limbaj este utilizat, cu foarte mici deosebiri, care nu sînt semnificative, de către toate cal-

culatearile de capacitate medie și mare.

Cuvintele folosite în limbajul FORTRAN alcătuiesc vocabularul limbajului.

Cuvintele și prepozițiile în limbajul FORTRAN se scriu după reguli precise a căror respectare este obligatorie.

3.2. Caractere FORTRAN

- litere majuscule ale alfabetului englez în număr de 26:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- cele 10 cifre ale sistemului zecimal de numerație;

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- caractere sau semne speciale:

= atribuire

. punct

+ plus

, virgulă

- minus

' apostrof

/ bară înclinată (paranteză stînga

* asterisc

) paranteză dreapta

⌘ preluată (and)

␣ sau b, blank sau spațiu alb.

- cuvinte cheie, constituind vocabularul fix:

un număr de cuvinte din limba engleză care nu pot fi modificate. Majoritatea din ele sînt prezentate în tabelul 3.1. în care sînt redată sensul în limba română și transcrierea fonetică a acestora.

Observații

1. pentru a nu se confunda litera "O" cu cifra zero se folosește scrierea barată a literei O, astfel: "Ø".

2. Folosirea cuvintelor cheie va fi făcută numai în sensul și cu destinația impusă de limbaj.

- cuvinte generate de programator, constituind vocabularul variabil al limbajului, respectînd regulile acestuia, sînt de două tipuri:

- nume simbolice sau identificatori

- etichete.

Nume simbolic sau identificator este o însușire de 1-8 caractere literare și/sau cifre în care primul caracter este INTOTDEAUNA o literă.

Observații

1. Depășirea numărului de 8 caractere la alcătuirea unui nume simbolic nu este considerată de către compilatorul FORTRAN

e creare, reținându-se în mod automat primele 8 din gir. În această situație poate apărea o coincidență de nume simbolice pentru două variabile diferite, ceea ce constituie, însă, o eroare gravă. De exemplu dacă TRANSALP și TRANSALPIN ar constitui pentru un programator două identificatori diferiți, pentru calculator va fi de fapt unul și același identificator. — TRANSALP.

Exemple corecte de identificatori

A	TUDOR	YQTX
B	MAMA	VQVAR
I	JL	III
I 2	JJ	

Exemple de cuvinte care nu sînt identificatori

A,17A	conține caracterul " , " străin definiției
B-C	conține caracterul "-" străin definiției
QT)A	conține caracterul ")" străin definiției
1A45	începe cu 1
VAL.T	conține caracterul " . " străin definiției

Se observă că un nume simbolic poate fi constituit din orice combinație de caractere alfanumerice cu singura condiție a respectării regulii de formare. În practică însă se folosesc, pentru a urmări mai ușor un program, identificatori astfel alcătuiți încît să sugereze domeniul de utilizare, practică ce ușurează scrierea și înțelegerea programului.

Etichetă este un număr fermat cu 1 pînă la 5 cifre, întreg fără semn, diferit de zero sau, altfel spus, un număr întreg N cuprins între limitele:

$$1 \leq N \leq 99999$$

Obs. Etichetele se folosesc pentru identificarea unor instrucțiuni și ca atare este interzisă utilizarea aceluiași etichete pentru două sau mai multe instrucțiuni. De altfel, o tentativă de acest fel este sancționată prin invalidarea programului în faza de compilare a acestuia.

Instrucțiunile limbajului FORTRAN sînt prepoziții constituite din cuvintele vocabularului.

Program e succesiune de instrucțiuni care descrie un algoritm.

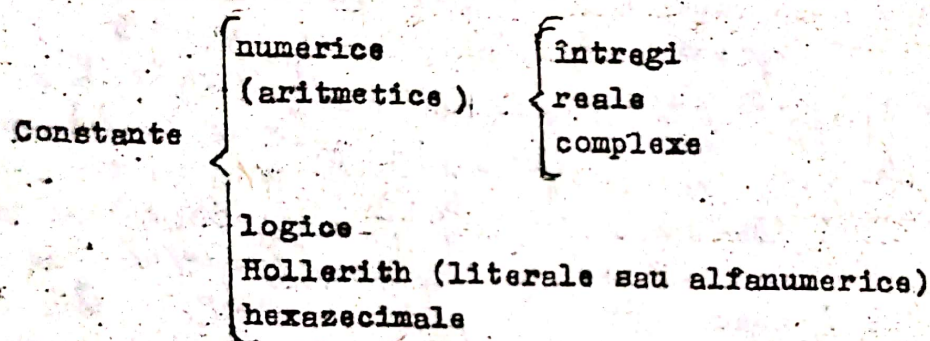
Tabel 3.1.

Cuvînt cheie	Semnificație	Transcriere fonetică
0	1	2
READ	citește	[ri:d]
WRITE	scrie	[rait]
GO TO	transfer la	[gou tu]
IF	dacă	[if]
DO	execută	[du]
REAL	real	[rial]
INTEGER	întreg	[intidzə]
COMPLEX	complex	[kəmpleks]
LOGICAL	logic	[lɒdʒikəl]
DOUBLE PRECISION	dublă precizie	[dʌbl prɪʒən]
CONTINUE	continuă	[kən'tɪnju:]
PRINT	tipărește, imprimă	[print]
STOP	oprește	[stop]
CALL	chama	[kɔ:l]
FUNCTION	funcție	[fʌŋksən]
COMPILE	a compila	[kəm'paɪl]
DIMENSION	dimensiune	[dɪmən'sən]
RETURN	întoarce	[ri'tə:n]
IMPLICIT	implicit	[ɪm'plɪsɪt]
COMMON	comun	[kɒmən]
DATA	date	[deɪtə]
EQUIVALENCE	echivalență, corespondență	[ɪ'kwɪvələns]
EXTERNAL	exterier	[ek'stə:nəl]
ENTRY	intrare, acces	[ɛntri]
DECODE	decodifică	[di:kəʊd]
ENCODE	codifică	[ɪn'kəʊd]
BUFFER	tampă	[bʌfə]
NAMelist	listă de nume	[neɪmlɪst]
PAUSE	pauză, întrerupere	[pɔ:z]
ASSIGN	asignare	[ə'saɪn]
NOT	nu	[nɒt]
AND	și	[ænd]
OR	sau, ori	[ɔ:r]
GREATER THAN	mai mare decât	[greɪtə 'ðen]
GREATER THAN OR	mai mare sau egal cu mai puțin ca	[greɪtə 'ðen ɔ:r]
EQUAL TO		[i:kwəl tu]
LESSTHAN		[les 'ðen]

0	1	2
LESS THAN OR	mai puțin sau	[les ðen ɔ:]
EQUAL	egal cu	[i:kwɔl tu]
EQUAL TO	egal cu	[i:kwɔl tu]
NOT EQUAL TO	neegal cu	[nɔt i:kwɔl tu]
DEFINE FILE	definește fișier	[dɪfaɪn faɪl]
SUBROUTINE	subrutină	[sabrʊtɪn]

3.3. Constante

Constantele se autodefinesc prin însăși apariția lor în program sau altfel spus sînt mărimi ale căror valori nu se modifică în cursul executării programului.



Constantă întregă: un șir de cifre zecimale, fără punct, cu sau fără semn.

Observații

- Se menține convenția din matematică privind scrierea opțională a semnului plus.
- Noțiunea de cifre zecimale se referă la baza în care se face scrierea numărului - baza 10.-.

Exemple de constante întregi:

0
29
.9993
-7342

Constante reale

Limbaajul FORTRAN admite trei tipuri de constante reale și anu-

me:

- constante reale de bază (CRB);
- constante reale de bază urmate de exponent zecimal (CRBEZ);
- constante întregi urmate de exponent zecimal (CIEZ);

CRB = un șir de cifre zecimale cu punct zecimal, cu sau fără semn.

Observație. Utilizarea noțiunii de punct zecimal, avînd semnificația de virgulă zecimală, se datorește faptului că în scrierea anglo-saxonă se folosește modul acesta de scriere.

Exponentul zecimal este reprezentat de litera E sau D, urmată de un grup de două cifre zecimale, cu sau fără semn. Litera E sau D reprezintă baza 10 de numerație.

Litera E indică în mod explicit prelucrarea în simplă precizie, iar litera D indică în mod explicit prelucrarea în dublă precizie a constantei respective.

Tabel 3.2.

Tipul constantei	Scriere FORTRAN	Echivalentul matematic
<u>CRB</u>	10.452	10,452
	-777.	-777,0
	0.1254	0,1254
	.83	0,83
	-7.23E+02	$-7,23 \times 10^2$
<u>CRBEZ</u>	33.4E-3	$33,4 \times 10^{-3}$
	99.4D4	$99,4 \times 10^4$
	-8.E04	$-8,0 \times 10^4$
	-8.E4	
	-8.E+04	
<u>CIEZ</u>	.7D-12	$0,7 \times 10^{-12}$
	7E4	7×10^4
	-888D-2	-888×10^{-2}

Exemple greșite de constante reale:

Tabel 3.3.

Forma greșită	Eroarea	Forma corectă
755,24	Virgula zecimală nu este admisă	755.24
374	Constantă întreagă lipsește punctul zecimal	374.
339.E-725	Depășește capacitatea exponentului	
7.3E+-7	două semne consecutive	7.3E-7 sau 7.3E+7
29.E1.2	exponentul nu este o constantă întreagă	29.E12

Constante complexe

O constantă complexă este reprezentată printr-o pereche ordonată de constante reale, cu sau fără semn, separate prin virgulă și închise între paranteze.

Prima constantă reprezintă partea reală a numărului complex, iar cea de-a doua - partea imaginară.

Exemple de constante complexe:

Tabel 3.4.

Scrierea FORTRAN	Echivalentul matematic
(7.4,-7.3)	$7,4-7,3i$
(11.,.5)	$11,0-0,5i$
(0.3E2,7.9E-2)	$0,3 \times 10^2 + 7,9 \times 10^{-2}i$

Pentru constantele reale și complexe este opțională prelucrarea în dublă precizie. Această modalitate de prelucrare se folosește atunci când se cer calculele cu o precizie ridicată.

Constante logice

O constantă logică poate avea una din valorile "adevărat" sau "fals", reprezentate în FORTRAN prin .TRUE. și respectiv .FALSE.

Constante Hollerith

Aceste constante se mai numesc în mod curent constante literale sau alfanumerice. S-a optat pentru a denumirea folosită în titlu pentru că, așa cum o arată definiția, denumirile amintite mai sus nu sînt suficiente și explicite pentru începători, putînd da naștere la confuzii.

O constantă Hollerith este un șir de caractere FORTRAN (litere și/sau cifre și/sau caractere speciale) cuprinse între apostrofuri sau precedate de descriptorul nH unde n este numărul de caractere ce urmează nemijlocit codul H.

Exemple

'FACULTATEA DE TEXTILE'
'X= ' 'RADACINA X= '
sau 22 H TABEL CU STUDENTII GR.
17 H VARIATIE CONTINUA

Obs.

1. Blancurile fiind caractere, vor fi considerate cînd se stabilește numărul de caractere din șir.
2. Existența în șir a unui apostrof impune dublarea lui în cazul transcrierii în FORTRAN. De exemplu expresia:
Anul '980 transcrie în FORTRAN arată astfel

' ANUL ''980'

Aceste constante sînt utilizate ca mesaje (asea deosebi de comentarii) afișate la imprimantă în zona de scriere a rezultatelor, pentru ușurarea înțelegerii activității programului sau pentru trasarea unor liniaturi ca tabele, sublinieri, curbe, etc.

Constante hexazecimale

O constantă hexazecimală este un șir de cifre hexazecimale precedat de descriptorul Z.

ZA7416BFD

3.4. Variabile

Variabila este o reprezentare simbolică a unei cantități ce ocupă un spațiu de memorie. Această definiție impune ca noțiunea de variabilă să fie întotdeauna în legătură biunivocă cu dubletul de noțiuni nume simbolic sau identificator și constantă, deoarece reprezentarea simbolică a unei variabile este asigurată de identificator, în timp ce spațiul de memorie este ocupat de o constantă.

$$v \leftrightarrow (n, c)$$

Variabila este de tipul pe care îl are constanta respectivă. Lungimea unei variabile = mărimea domeniului de memorie repartizat unei variabile și se exprimă în octeți.

Lungimea unei variabile depinde de tipul ei, putând avea o lungime standard și o lungime opțională conform tabelului:

Tabel 3

Tipul variabilei	Lungimea standard	Lungimea opțională
Intreg	4	8
Real simplă precizie	4	8
Real dublă precizie	8	-
Complex	8	16
Logic	4	8

Lungimile opționale se folosesc atunci când se cere o precizie mai mare a rezultatelor.

Stabilirea tipului unei variabile se poate face folosind una din următoarele trei moduri de specificare a tipului:

1. Convenția FORTRAN predefinită de declarare a tipului;
2. Specificarea implicită a tipului variabilei

3. Specificarea explicită a tipului variabilei

Convenția FORTRAN predefinită : toate variabilele ale căror nume simbolice încep cu una din literele I, J, K, L, M, N, sunt considerate ca fiind de tip întreg; variabilele ale căror nume simbolice încep cu alte litere decât cele menționate mai sus, sunt considerate ca fiind de tip real.

Exemple de nume de variabile întregi :

KOL	K	IQTK
ML	KI	NAMA
NN	KK	JIU

Exemple de nume de variabile reale:

A	ABRA	TIMP
AI	VAL2	QITO
AB	Q4TI	WI

In ce privește modurile de specificare explicit și implicit, acestea vor fi discutate pe larg în cap.6.

Obs. Convenția FORTRAN se referă la variabile de lungime standard și acționează când nu se fac declarații de sens contrar.

3.5. Tablou FORTRAN

Se numește tablou o mulțime de date de același tip identificate prin același nume simbolic. Apelarea unei date din tablou se face scriind numele tabloului însoțit, între paranteze, de un grup de 1 la 7 indici separați prin virgulă, ale căror valori indică poziția datei respective în tablou. Numărul de elemente ale tabloului este dat de produsul valorilor maxime ale indicilor.

Indicii sînt constante întregi, variabile întregi sau expresii aritmetice de tip întreg.

Indicii în FORTRAN trebuie să fie întregi, pozitivi și diferiți de zero.

Exemple de tablouri FORTRAN:

A(100)-tablou unidimensional A cu 100 de elemente;
 B(3,5)-tablou bidimensional (matrice) cu 3x5 elemente;
 TAB(7,9,10)-tablou tridimensional cu 7x9x10 elemente;
 I(J,K+4)-tablou bidimensional cu Jx(K+4) elemente.

Obs.

1. Tipul tabloului se stabilește, ca și pentru variabile, prin numele simbolic al acestuia.

2. Mărimea tabloului se indică calculatorului printr-o instrucțiune specializată.

3. Folosirea unor variabile sau expresii drept indici impune ca în momentul referirii la tablou aceste variabile să fie deja definite sau aceste expresii să fie deja evaluate.

Indiferent de forma externă a tabloului în memorie, acesta

se aranjează sub forma unui şir în baza regulei indicilor, care poate avea una din următoarele două forme:

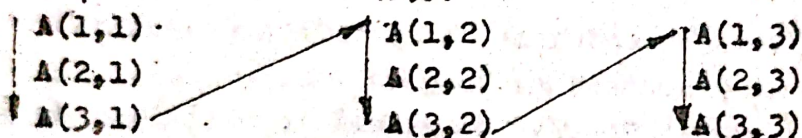
Regula indicilor implicită:

Primul indice creşte cel mai repede, ultimul creşte cel mai încet.

Regula indicilor explicită:

Indicele cel mai interior creşte cel mai repede, indicele cel mai exterior creşte cel mai încet.

De exemplu, aplicarea regulei indicilor implicită în cazul unui tablou bidimensional $A(3,3)$



cu termenul generic $A(I,J)$, primul indice este I , al doilea şi ultimul indice este J . Indicele I ia valori de la 1 la 3, în timp ce J ia valoarea 1. Deci $A(1,1)$, $A(2,1)$, $A(3,1)$.

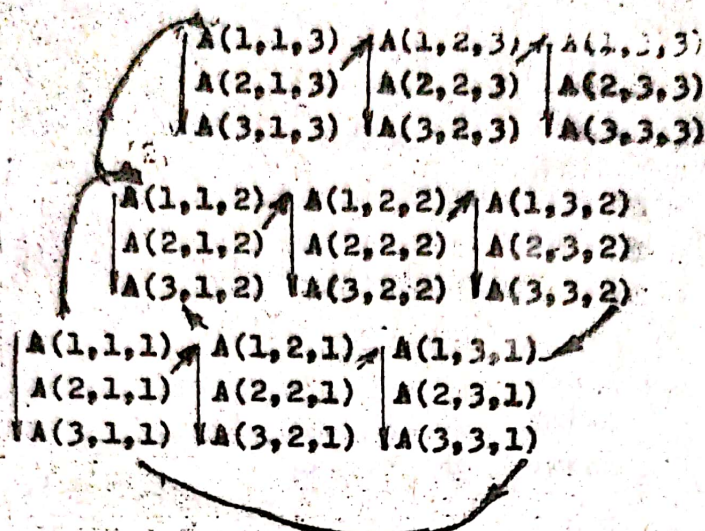
Ajungând indicele I la valoarea maximă, indicele J va căpăta următoarea valoare, adică 2, moment în care I va pleca din nou de la valoarea iniţială până la cea finală, adică $A(1,2)$, $A(2,2)$, $A(3,2)$.

În final $J = 3$, iar I va varia din nou între valoarea iniţială şi finală adică $A(1,3)$, $A(2,3)$, $A(3,3)$. Se observă din schema

de mai sus că, în virtutea acestei reguli, în memorie se aşează elementele tabloului după coloane. Deci, în momentul înscrierii datelor pe formular, trebuie avută în vedere această observaţie.

În caz contrar, transcriind datele după linie pe cartela de date (aşa cum sîntem obișnuiți a scrie de regulă o matrice), în memoria calculatorului datele vor fi tratate ca fiind pe coloane şi, menţinînd exemplul cu matricea, înseamnă că în memorie se depozitează transpusa acesteia.

Pentru un tablou cu trei dimensiuni, prin aplicarea acestei reguli pentru depunerea în memorie valorile vor fi citite după schema din figura următoare:



Obs. Regula este valabilă în lipsa unor indicații de sens contrar date calculatorului prin varianta explicită a regulii.

De exemplu, dacă se dorește introducerea în memorie după linie a elementelor tabloului A , utilizând regula explicită a indicilor, comanda este următoarea:

$((A(I, J), J = 1, 3), I = 1, 3)$

Se observă că în cea mai interioară paranteză se găsește indicele J , indicele de coloană, care va varia cel mai repede, parcurgând toate valorile sale, pentru fiecare valoare a lui I (indicele de linie), astfel că în memorie se vor așeza elementele tabloului A în succesiunea $A(1,1)$, $A(1,2)$, $A(1,3)$, $A(2,1)$, $A(2,2)$, $A(2,3)$, $A(3,1)$, $A(3,2)$, $A(3,3)$.

Tabloul A scris	$A(1,1)$	$A(1,2)$	$A(1,3)$
în forma obișnuită	$A(2,1)$	$A(2,2)$	$A(2,3)$
	$A(3,1)$	$A(3,2)$	$A(3,3)$

Schematic, exemplele de mai sus se prezintă astfel:

Declarație de dimensiune $\text{DIMENSION } A(3,3)$

Element generic $A(I, J)$

Ordinea așezării în
memorie a elementelor
tabloului A

A(1,1)
A(2,1)
A(3,1) _____ coloana 1
A(1,2)
A(2,2)
A(3,2) _____ coloana 2
A(1,3)
A(2,3)
A(3,3) _____ coloana 3

primul indice
variază mai repede

Regula implicită a indicilor pentru tabloul A(3,3)

Tabloul A scris în
forma obișnuită

A(1,1) A(1,2) A(1,3)
A(2,1) A(2,2) A(2,3)
A(3,1) A(3,2) A(3,3)

Declarația de dimensiune

DIMENSION A(3,3)

Element generic

((A(I,J), J=1,3), I=1,3)

Ordinea așezării
în memorie a elementelor
tabloului A.

↓ indicele
A(1,1) al 2-lea variază
A(1,2) mai rapid
A(1,3) _____ linia 1-a
A(2,1)
A(2,2)
A(2,3) _____ linia 2-a
A(3,1)
A(3,2)
A(3,3) _____ linia 3-a

Regula explicită a indicilor pentru tabloul A(3,3).

3.6. Expresii

Varianta FORTRAN IV acceptă două tipuri de expresii:

- expresii aritmetice;
- expresii logice.

3.6.1. Expresii aritmetice

Expresiile aritmetice sînt de două tipuri:

- a. expresii aritmetice elementare (EAE)
 - b. expresii aritmetice propriu-zise (EAP)
- a. EAE poate fi formată dintr-o constantă, o variabilă, o variabilă cu indici, o referință la o funcție, sau o altă expresie închisă între paranteze.

Exemple de EAE

24942

-9.334

72.72D-14

(1429.2, -742.2)

X(3,4)

CGS(ALFA)

b. EAP sînt formate din EAE legate între ele prin operatori aritmetici (folosind eventual și paranteze).

Operatori aritmetici în FORTRAN sînt:

- " + " adunare
- " - " scădere
- " * " înmulțire
- " ** " ridicare la putere.

Expresiile: A + B înseamnă A plus B

A - B înseamnă A minus B

A/B înseamnă A împărțit la B

A * B înseamnă A înmulțit cu B

A ** B înseamnă A ridicat la puterea B.

Calculule indicate într-o expresie se execută respectînd următoarea ordine de prioritate privind efectuarea operațiilor:

- 1) evaluarea funcțiilor
- 2) ridicarea la putere
- 3) înmulțirea și împărțirea
- 4) adunarea și scăderea.

La priorități egale operațiile se execută de la stînga la dreapta cu excepția ridicării la putere, care se execută de la dreapta la stînga.

Prin introducerea parantezelor se impune o anumită ordine de evaluare.

Observație. Folosiți parantezele la transcrierea în FORTRAN a expresiilor aritmetice pentru păstrarea nealterată a sensului acestora.

Astfel expresia $\frac{a}{b \cdot c}$ se transcrie $A/(B * C)$ și nu $A/B * C$ care înseamnă de fapt $\frac{a}{b} \cdot c$, ceea ce este evident altă expresie.

Exemple de expresii aritmetice propriu-zise:

FORMA FORTRAN

ECHIVALENTUL MATEMATIC

$Y + 75.2 * X - \sin(\text{ALFA})$

$Y + 75,2x - \sin(\alpha)$

$(A + B/C - \text{SQRT}(B))/C$

$a + b/c - \sqrt{b}$

c

$4 * B * \text{ALOG10}(X) - A * 24$

$4b \log(x) - a^4$

Exemple de expresii aritmetice cu erori:

- $A + BC$ între B și C lipsește operatorul $*$
- $A/\text{ALOG}(B) - C)/D$ lipsește paranteza stînga
- $(C + D)A + B$ lipsește operatorul aritmetic din fața variabilei A
- $7.5 - + A * 3$ doi operatori aritmetici nu se pot urma nemijlocit

Deoarece operanzii unei EA pot fi întregi (I), real simplă precizie (RS), real dublă precizie (RD), complex simplă precizie (CS), complex dublă precizie (CD), rezultatul evaluării unei expresii va avea tipul conform tabelului următor pe baza ordinii de prioritate din tabel 3.7.

Tabel 3.6.

Operatorul +, -, /, *	Operandul 2				
	I	RS	RD	CS	CD
I	I	RS	RD	CS	CD
RS	RS	RS	RD	CS	CD
RD	RD	RD	RD	CD	CD
CS	CS	CS	CD	CS	CD
CD	CD	CD	CD	CD	CD

Tabel 3.7

tip operand	Ordinea de prioritate
CD	1
CS	2
RD	3
RS	4
I	5

Obs. La transcrierea în FORTRAN a unei expresii aritmetice se cere o mare atenție pentru a evita anomaliile datorate convertirilor.

Astfel rezultatul expresiei aritmetice

$$3/2 = 1$$

evaluată pe calculator, este 0 deoarece împărțirea a doi operanzi întregi, 3 și respectiv 2, dă un rezultat întreg, adică 1 și nu 1,5 cum e normal. Pentru obținerea rezultatului corect matematic expresia trebuie scrisă astfel:

$$3./2 = 1 \text{ sau } 3/2.-1 \text{ sau } 3./2. = 1$$

caz în care rezultatul este corect, adică 1,5.

Exemplu:

Expresia

$$B = C/D + T \times \times 2/Q$$

se va evalua ca mai jos:

$$\begin{array}{c} B = C/D + T \times \times 2/Q \\ \underbrace{\quad\quad\quad}_{V_2} \quad \underbrace{\quad\quad\quad}_{V_1} \\ \underbrace{\quad\quad\quad}_{V_4} \quad \underbrace{\quad\quad\quad}_{V_3} \\ \underbrace{\quad\quad\quad\quad\quad\quad\quad}_{R} \end{array}$$

Expresia

$$M \times A/B \times \times J$$

în care M și J sînt variabile întregi de lungime 4, A este variabilă reală de lungime 4, B este variabilă reală de lungime 8, dă un rezultat R, o variabilă reală de lungime 8, ca mai jos:

$$\begin{array}{c} M \times A/B \times \times J \\ \underbrace{\quad\quad\quad}_{V} \quad \underbrace{\quad\quad\quad}_{\text{variabilă reală de lungime 4}} \\ \underbrace{\quad\quad\quad}_{\text{variabilă reală de lungime 8}} \\ \underbrace{\quad\quad\quad\quad\quad\quad\quad}_{R \quad \text{variabilă reală de lungime 8}} \end{array}$$

Funcțiile FORTRAN sînt prezentate în tabelele 3.8 și 3.9.

Se observă din tabelele 3.8 și 3.9 că aceste funcții au nume care indică problema ^{pe} care o rezolvă. Se recomandă ca la utilizarea unei astfel de funcții să se consulte cu atenție manualul, pentru a folosi corect atât numele cit și tipul și numărul argumentelor; în caz contrar programul este invalidat.

Funcții intrinseci (tab.3.8)

Nr. crt.	Funcția Numele	Tipul	Parametrii Tipul	Exemplu de Numărul utilizare	Utilitatea	
1.	ABS	R	R	1	ABS(3.14159)	Calculul valorii absolute a parametrului
	IABS	I	I	1	IABS(I=J-14)	
	DABS	D	D	1		
2.	AIMT	R	R	1	AIMT(B)	Rotunjirea valorii parametrului la unitatea inferioară în valoare absolută.
	INT	I	R	1	INT(X=2-A/B)	
	IDINT	I	D	1	IDINT(X)	
3.	AMOD	R	R	2	AMOD(X,Y)	Restul împărțirii întregi a celor 2 parametri
	MOD	I	I	2		
4.	AMAXO	R	I	2	AMAXO(IJ,4)	Găsirea valorii maxime a parametrilor specificați
	AMAXI	R	R	2	AMAXI(X,3,I)	
	MAXO	I	I	2	MAXO(I,J,4,3=K)	
	MAXI	I	R	2	MAXI(A,3.)	
	DAMAXI	D	D	2	DAMAXI(A,B=C)	
5.	AMINO	R	I	2	AMINO(K1,K2)	Calculul minimumului dintre parametrii specificați
	AMINI	R	R	2	AMINI(R,T,U,V)	
	MINO	I	I	2	MINO(1,M,I=J)	
	MINI	I	R	2	MINI(A)	
	DMINI	D	D	2	DMINI(D=C)	
6.	FLOAT	R	I	1	FLOAT(3)	Conversia din întreg în real
7.	DFLOAT	D	I	1	DFLOAT(4)	Conversie din întreg în dublă precizie
8.	IFIX	I	R	1	IFIX(75,2)	Conversie din real în întreg
9.	DIM	R	R	2	DIM(A,B)	Calculează diferența dintre primul parametru și cel mai mic dintre ei
	IDIM	I	I	2	IDIM(I,3)	
10.	SNGL			1	SNGL(R)	Partea cea mai semnificativă a parametrului e convertită în real

Continuare Tabel 3.8

11. SIGN	R	R	2	SIGN(-3,5,+1)	Transfer de semn (valoarea absolută a primului parametru la semnul celui de al doilea)
ISIGN	I	I	2	ISIGN(-3,+4)	
DSIGN	D	D	2	DSIGN(A,B1)	
12. REAL	R	C	1	REAL(2)	Calculul părții reale a numărului complex
13. AIMAG	R	C	1	AIMAG(	Calculul părții imaginare

Funcții externe uzuale (tab.3.9)

Nr.	Funcția	Parametrii	Exemplu	Utilitatea
crt.	Numele	Tipul	Tipul Numărul	
1.	E2XP	R	R 1	EXP(X) Calculul funcției exponen-
	DEXP	D	D 1	DEXP(X) tiale
	CEXP	C	C 1	CEXP(X)
2.	ALOG	R	R 1	ALOG(A+B) Calculul logaritmului ze-
	DLOG	D	D 1	DLOG(3=A) cimal
	CLOG	C	C 1	CLOG(B)
3.	ALOG 10	R	R 1	ALOG(X) Calculul logaritmului ze-
	DLOG 10	D	D 1	DLOG(X) cimal
4.	SIN	R	R 1	SIN(X) Funcția sinus
	DSIN	D	D 1	DSIN(X)
	CSIN	C	C 1	CSIN(X)
5.	COS	R	R 1	COS(X) Funcția cosinus
	DCOS	D	D 1	DCOS(X)
	CCOS	C	C 1	CCOS(X)
6.	SINH	R	R 1	SINH(B) Funcția sinus hiperbolică
	DSINH	D	D 1	DSINH(A)
7.	COSH	R	R 1	COSH(X-3) Funcția cosinus hiperbolică
	DCOSH	D	D 1	DCOSH(Y)
8.	TANH	R	R 1	TANH(A) Funcția tangentă hiperbolică
	DTANH	D	D 1	DTANH(B)
9.	TANH	R	R 1	TANH(X) Tangentă hiperbolică
10.	SQRT	R	R 1	SQRT(X) Calculul rădăcinii pătrate
	DSQRT	D	D 1	DSQRT(X)
	CSQRT	C	C 1	CSQRT(X)
11.	ATAN	R	R 1	ATAN(X) Funcția arctg(x)
	DTAN	D	D 1	DTAN(X)

Continuare Tabel 3.9

12.	ATAN 2	R	R	2	ATAN 2(X,Y) Funcția arctg x/y
	DTAN 2	D	D	2	DTAN 2(X,Y)
13.	DMOD	D	D	2	DMOD (A,B) Restul lui A modulo B
14.	CABS	R	C	1	CABS(W) Modulul unui număr complex

3.6.2. Expresii logice

Limbajul FORTAN prevede utilizarea a 3 tipuri de expresii logice;

a. expresii logice elementare (ELE)

b. expresii relaționale (ER)

c. expresii logice propriu-zise (ELP)

a. ELE sînt formate dintr-o constantă logică, o variabilă logică, o funcție logică, expresii logice închise între paranteze.

b. ER sînt formate din EAE sau EAP de tip întreg sau real, legate între ele prin operatori de relație sau relaționali.

c. ELP sînt formate din ELE sau ELP, sau ELE și ELP legate între ele prin operatori logici.

Operatori raționali sînt:

Tabel 3.10

Nr. crt.	Operatorul relational	Codificarea operatorului	Denumirea operatorului
1.	<	.LT.	Less Than
2.	≤	.LE.	Less Equal
3.	=	.EQ.	Equal.
4.	≥	.GE.	Greater Equal
5.	>	.GT.	Greater
6.	≠	.NE.	Not Equal

Operatorii logici sînt:

.NOT. - negația "nu"

.AND. - conjuncție "și"

.OR. - disjuncție "sau"

Se operează cu operatorii logici pe baza tabelelor de adevăr:

Tabelul .NOT.

X	.NOT.X
.TRUE.	.FALSE.
.FALSE.	.TRUE.

Tabelul .AND. și .OR.

X	Y	X.AND.Y	X.OR.Y
T	F	F	T
T	T	T	T
F	F	F	F
F	T	F	T

Exemple de .BLE

.TRUE.

.FALSE.

LOG - variabilă logică declarată ca atare.

Exemple de .ER

(A + B).GT.0

(A * 2 - 4*ABC).GE.0

Exemple de .ELP

X.AND.Y.

(A + B).GT.C.NOT.A.

(A**2 - B**2).EQ.D.AND.LOG

Evaluarea expresiilor logice:

Evaluarea unei expresii logice în forma ei cea mai generală comportă următoarele etape:

- se evaluează EA ce compun ER;
- se evaluează ER, rezultatul acestora putând fi .TRUE. sau .FALSE.

- evaluarea expresiilor logice din paranteze

- evaluarea de la stînga la dreapta pentru priorități egale a operatorilor logici în următoarea ordine:

.NOT,

.AND.

.OR.

Rezultatul este o valoare logică .TRUE. sau .FALSE.

Observație

- doi operatori logici nu se pot urma nemijlocit într-o expresie, decât dacă al doilea este .NOT.

Corect. A.AND..NOT.B

Inc corect A.OR..AND.B

Exemple:

Fie expresia logică:

A.OR.B.AND.C.AND..NOT.D

in care variabilele logice A și C au valoarea logică .FALSE.
iar B și D au valoarea logică .TRUE.

A.OR.B.AND.C.AND. .NOT. D

_____	N	_____	are valoare .FALSE.
_____	M	_____	are valoare .FALSE.
_____	P	_____	are valoare .FALSE.
_____	R	_____	are valoare .FALSE.

Fie expresia logică;

A.GT.B.OR.C.NE.D.AND.E.EQ.F

in care A,B,C,D,E,F sînt variabile întregi cu valorile 1,2,3, 18,5,10. Se cere valoarea expresiei;

_____	U	_____	.FALSE.
_____	V	_____	.TRUE.
_____	T	_____	.FALSE.
_____	W	_____	.FALSE.
_____	R	_____	.FALSE.

3.7. Instrucțiunile limbajului FORTRAN

3.7.1. Clasificarea:

Instrucțiunile limbajului FORTRAN sînt de două tipuri:

- instrucțiuni operante sau executabile;
- instrucțiuni inoperante sau neexecutabile

Instrucțiunile operante sînt instrucțiuni care indică efectuarea unei anumite operații.

Instrucțiunile inoperante sînt instrucțiunile care se adresează compilatorului și care dau informații necesare rezolvării programului.

Instrucțiunile operante sînt clasificate în funcție de operația pe care o indică:

- instrucțiuni de intrare-ieșire a datelor;
- instrucțiuni de atribuire;
- instrucțiuni de salt;
- instrucțiuni de ciclare;
- instrucțiuni de procedură.

Instrucțiunile inoperante, în funcție de rolul pe care îl au, sînt de următoarele tipuri:

- declarații de tip, care specifică tipul variabilelor;
- declarații de alocare, care asigură organizarea datelor în memorie;
- specificații de FORMAT, care indică modul de așezare a datelor pe înregistrări;
- instrucțiuni de definire a procedurilor, care indică numele și parametrii formali ai acestora.

Instrucțiunile operante traduc în limbajul FORTRAN blocurile corespunzătoare din schema logică.

Instrucțiunile inoperante nu au echivalente blocuri în schemele logice.

În tabelul 3.11 sînt prezentate cuvintele cheie prin care se identifică, de regulă, instrucțiunea și operația efectuată sau indicația dată.

Tabelul 3.11

Instrucțiuni operante		Instrucțiuni inoperante	
Denumire	Operația	Denumire	Indicația
		BLOCK DATA	inițializează valori în proceduri
ASSIGN	asignare	COMMON	comun
CALL	chemare	COMPLEX	complex
CONTINUE	continuă	DATA	data
DO	execută	DIMENSION	dimensiune
END	sfîrșit	DOUBLE PRECISION	dublă precizie
GOTO	transfer la	ENTRY	intrare
IF	dacă	EQUIVALENCE	echivalent
INSTR.DE ATRIBUIRE	calcul	EXTERNAL	extern
PAUSE	întrerupere	FUNCTION	funcție
PRINT	imprimă	INSTR.DE DEFINIRE A FUNCȚIILOR	
PUNCH	perforează	IMPLICIT	implicit
READ	citește	INTEGER	întreg

Continuare tabel 3.11

RETURN	întoarcere	LOGICAL	logic
STOP	oprește	NAMelist	nume de listă
WRITE	scrie	REAL	real
		SUBROUTINE	subrutină

3.7.2. Structura unui program scris în limbajul FORTRAN

Programele scrise în limbajul FORTRAN trebuie să prezinte o anumită structură, respectiv o anumită ordine de așezare a instrucțiunilor. În componența unui program pot să apară mai multe unități de program și anume un program principal și unul sau mai multe subprograme. Frecvent însă programul este format dintr-o singură unitate și anume programul principal. Noțiunea de subprogram indică raportul de subordonare funcțională, subprogramul oferind programului apelant cel puțin o valoare.

Instrucțiunile ce compun un program FORTRAN se vor așeza într-o anumită ordine pe nivele:

nivel 1 - instrucțiuni de specificare a tipului sau subprogramului

nivel 2 - este zona de declarare

nivel 3 - instrucțiunea de inițializare

nivel 4 - funcții formulă

nivel 5 - instrucțiuni executabile

nivel 6 - instrucțiuni END

Observații - liniile de comentariu pot să apară oriunde în program,

- instrucțiunile FORMAT ^{pot} să apară oriunde în nivelele 2, 3, 4, 5,
 - nivelele 5 și 6 sînt obligatorii, adică trebuie să existe cel puțin o instrucțiune executabilă și instrucțiunea de sfîrșit fizic al programului END,
 - programele principale nu au instrucțiuni de nivel 1.
- Apariția ei arată că este vorba de un subprogram (SUBROUTINE, FUNCTION, sau BLOCK DATA)

3.7.3 Formularul de programare FORTRAN

Instrucțiunile unui program se înscriu într-un formular special tipărit pe format A4 avînd 80 de coloane și 25 de linii. Cele 80 de coloane se împart în 4 zone și anume:

- D - Zonă marcată: col 73-80

[illegible]

Fig. Formularul de programare FORTRAN

In zona A se înscriu etichetele cu ajutorul cărora se identifică instrucțiunile.

In zona B se înscrie caracterul de continuare ce se constituie din orice caracter ~~FORT~~TRAN, cu excepția caracterelor zero sau blanc. Acest caracter indică compilatorului că c e s a e e urmează face parte din instrucțiunea de pe linia precedentă.

In zona C se înscriu instrucțiunile programului sursă, de regulă o instrucțiune pe fiecare linie. Dacă nu este de ajuns se trece pe linia următoare, după ce în prealabil pe coloana 6 s-a înscris caracterul de continuare.

O instrucțiune se poate continua pe maximum 19 linii a căror zonă de etichetă trebuie să fie liberă.

Caracterele instrucțiunii se pot scrie unele după altele sau cu spații între ele.

De exemplu sînt echivalente următoarele două moduri de înscriere în formulare:

```

1 READ(105,14)I,J,K
  R E A D(105, 1 ) I, J, K

```

recomandându-se prima formă atât pentru timpul necesar transcrierii cât și pentru estetica textului.

Excepție fac constantele Hollerith la care spațiul liber se consideră caracter.

Zona D este ignorată de compilator și este folosită pentru

numerotarea cartelilor sau pentru unele marcaje necesare programatorului.

Informații înscrise din greșeală în această zonă sînt pierdute pentru program, cu consecințele corespunzătoare.

În unele cazuri, mai ales pentru programatorii începători, se înserează între diferite secvențe de program explicații numite comentarii. Aceste comentarii se înscriu din orice coloană a formularului, după ce pe prima coloană s-a înscris caracterul C. Datele, dacă există, se înscriu începînd din coloana 1 pînă în coloana 80, în conformitate cu indicațiile FORMATULUI asociat.

Obs. Inscrierea instrucțiunilor pe formular se va face cu mare acuratețe; fiecărui caracter i se rezervă o singură căsuță. De aici pericolul interpretării dubioase a caracterelor scrise neglijent. Astfel se pot confunda caracterele N cu M, „cu”, 5 cu S, 7 cu Z, I cu J, etc.

După ce s-au înscris pe formular instrucțiunile, formularul este trecut la perforatorul de cartele, la care, fiecare linie de pe formular se transformă într-o cartelă. Se obține astfel un pachet de cartele ca în figura 3.1.

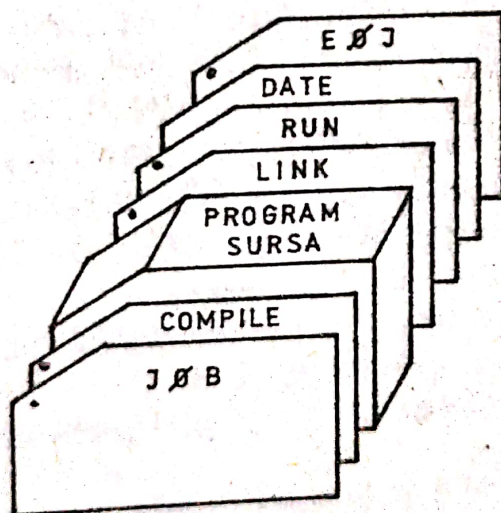


Fig.3.1. - Pachet de cartele

3.7.4. Cartele de comandă.

3.7.4.1. Fazele de rezolvare a unui program FORTRAN pe calculator.

Aceste faze sînt:

- compilarea
- editarea de legături
- execuția

În faza de compilare are loc traducerea programului citit de pe cartele, numit program sursă, în limbajul intern al calculatorului, fază în care se face și verificarea corectitudinii sintactice a propozițiilor. Dacă apar erori în această fază, în funcție de gravitatea lor (v. cap. 11) programul poate trece la etapa următoare sau este invalidat.

Invalidarea se face prin listarea programului și a instrucțiunilor în care au apărut erori, iar sub aceste instrucțiuni sînt tipărite mesaje de eroare adecvate.

În urma compilării se generează programul obiect în formă BT (binar translatabilă).

Programul obiect este tradus în formă IMT (imagine memorie translatabilă) în faza de editare a legăturilor.

În faza de execuție se încarcă în memorie programul IMT, se citesc datele și are loc prelucrarea acestora în urma căreia se arîșează rezultatele.

Toate aceste faze se afîșează automat sub controlul sistemului de operare, pe baza cartelelor de comandă specifice fiecărui calculator.

Cartelele de comandă, numite și cartele cu punct deoarece au punct în coloana 1, bordează programul sursă, ca în figura 3.1.

3.7.4.2. Cartelele de comandă.

În cele ce urmează sînt prezentate numai cartelele de comandă așa numite de "nivel 1", care se folosesc pentru execuția unor programe care nu au nevoie de subprograme din bibliotecă.

Cartela JOB este prima cartelă a programului și are forma:

1	11
.	JOB nume1,AN: nume2,PN: nume3

în care:

JOB-specifică începutul unui nou program

nume 1-numele programului și se constituie din maximum 8 caractere alfanumerice, primul caracter fiind obligatoriu o literă;

nume 2-codul contului plătitor și se constituie din 4 caractere alfanumerice,

nume 3-numele programatorului în clar sau codificat, format din până la 8 caractere alfanumerice.

Structura cartelei JOB trebuie respectată necondiționat.

Nerespectarea structurii acestuia duce la invalidarea programului în faza aceasta (v. cap. 11).

Cartela COMPILE.

Urmează nemijlocit cartelei JOB și are forma:

1	11
.	COMPILE FORTRAN opțiuni

Această cartelă aduce în memoria internă programul compilator FORTRAN și execută compilarea cu:

- listarea programului sursă
- transferă programul obiect pe disc
- listează lungimea modulelor de program obiect
- listează instrucțiunile în care s-au găsit erori sintactice (cu indicarea locului erorii și un mesaj adecvat)
- listează numărul total de erori și nivelul maxim de eroare întâlnit.

Obs. Instrucțiunea este considerată greșită la prima eroare întâlnită de la stînga la dreapta. Deci în cazul semnalării unei erori se recomandă ca, odată cu tratarea acesteia, să se verifice corectitudinea sintactică a întregii instrucțiuni.

Opțiunile permit obținerea unor informații suplimentare. Pentru detalii vezi bibliografia.

Cartela LINK.

Această cartelă se asează imediat după instrucțiunea END a programului sursă. Această cartelă pune în lucru editorul de legături care va genera forma LMT a programului.

Forma cartelei LINK este:

1	11
.	LINK opțiuni

Opțiunile permit obținerea unor informații suplimentare.

Cartela RUN.

Această cartelă, așezată după cartela LINK, are ca efect trans-

ferul programului în forma INT în memoria operativă și declanșarea execuției acestuia.

Are forma:

1	11
.	RUN NL:nr1,TIME:nr2,AD:a1,a2

Cele mai uzitate opțiuni sînt:

-NL:nr.1 -număr de linii și arată în mod explicit numărul de linii maxim de imprimat. Dacă lipsește se consideră implicit NL=1000 linii.

-TIME:nr.2-timpul în minute afectat execuției programului. Dacă lipsește se consideră implicit 3 minute.

-AD:a₁,a₂-specifică listarea la imprimantă a memoriei în cazul unei erori de la adresa a₁ la adresa a₂. Dacă lipsește se listează automat întregul conținut al memoriei interne. În practica didactică se indică a se utiliza opțiunea AD:0,0 ceea ce are ca efect interzicerea listării memoriei.

Cartela EOJ.

Aceasta indică sfîrșitul Jobului și are forma:

1	11
.	EOJ

În practică, un program poate trece prin toate cele trei faze în etape sau dintr-o dată, cazuri în care structura Job-ului va fi:

-în cazul compilării:

1	5 6 7	11
.		JOB
.		COMPILE
.		PROGRAM SURSA
.		-EOJ

-în cazul compilării și editării de legături:

1	5 6 7	11
.		JOB
.		COMPILE
.		PROGRAM SURSA
.		LINK
.		EOJ

1 3 7 11
JOB LANT
COMPILE FLAG BJH
% COMPILE FLAG

PROGRAM SURSA 1
END

% RUN
Date(dacă există)

% COMPILE FLAG
PROGRAM SURSA
END

% RUN
Date(dacă există)
EOJ

Aceasta este structura unui program în lanț.

Obs. Cartelele cu punct se montează de către Oficiul relații
al Centralnei de calcul.

CAPITOLUL 4

INSTRUCȚIUNI DE INTRARE-IEȘIRE

4.1. Forma generală a instrucțiunii intrare-ieșire

Instrucțiunile de intrare/ieșire comandă efectuarea transferului de date înspre și dinspre calculator.

Ele conțin indicații privind operația ce se execută, perifericul utilizat, variabilele implicate, etc.

Forma generală a unei instrucțiuni de intrare/ieșire este:

$$\parallel Cv(n, et) \text{ listă}$$

în care: - Cv = cuvânt cheie care indică operația ce urmează a se executa;

- n = numărul FORTRAN al perifericului sau număr simbolic al perifericului implicat în transferul de date.

- et = eticheta instrucțiunii FORMAT cu ajutorul căreia (prin indicațiile pe care le dă) se execută operația de intrare/ieșire;

- listă = o succesiune de nume de variabile, despărțite prin virgulă, a căror valori urmează a fi transferate pe/de pe înregistrare.

După ultimul nume simbolic nu se scrie virgulă.

Inregistrare: unitatea de bază a fișierului de date. Este, după caz, o cartelă sau un rând la imprimantă având în vedere

necesitățile cursului de față, în care se face referirea la fișierele organizate pe cartele perforate sau pe imprimantă. Fiecare înregistrare este divizată în câmpuri sau zone (coloare la cartela perforată, spații la imprimantă).

4.2. Instrucțiuni de intrare-ieșire standard

4.2.1. Instrucțiunea de intrare (citire).

Are forma generală:

$[et] \parallel \text{READ}(n, et) \text{ listă}$

în care: READ = cuvânt cheie cu sensul "CITEȘTE"

n = numărul FORTRAN al unității de intrare (cititorul de cartele n=105);

listă = o succesiune de nume de variabile, tablouri despărțite prin virgulă;

et₁ = etichetă opțională;

et = eticheta instrucțiunii FORMAT atagată.

Interpretarea instrucțiunii

CITEȘTE DIN SETUL SAU FIȘIERUL DE DATE n, CONFORM CU INDICAȚIILE DATE DE INSTRUCȚIUNEA CU ETICHETA et VALORIILE VARIABILELOR DIN listă.

În urma efectuării acestei instrucțiuni în memoria internă a calculatorului se depun în locațiile acordate variabilelor din listă, valorile corespunzătoare.

Exemplu:

5	6	7	READ(105, 10) A, B, I, KP(2)
10			FORMAT(-----)

Interpretare. Se citește (READ) de pe cartele perforate (105) conform indicațiilor date de instrucțiunea FORMAT cu eticheta 10, valorile variabilelor reale A și B, a variabilei întregi I și a elementului al doilea al tabloului KP.

4.2.2. Instrucțiunea de ieșire (scriere)

Are forma generală:

$[et_1] \parallel \text{WRITE}(n, et) \text{ listă}$

în care: WRITE = cuvânt cheie / cu sensul de "SCRIE";

n = numărul perifericului utilizat în operația de scriere, n = 108 pentru imprimantă;

listă = o înșiruire de indentificatori ai variabilelor ale căror valori se vor scrie la imprimantă;

et₁ = etichetă opțională;

et = eticheta instrucțiunii FFORMAT atasate.

Obs. Dacă nu se tipăresc valori, lista de variabile este vidă.

Interpretare. SCRIE LA IMPRIMANTA (n) CONFORM INDICAȚIILOR DATE DE INSTRUCȚIUNEA FFORMAT CU ETICHETA et VALORILE VARIABILELOR DIN listă.

În urma efectuării acestei instrucțiuni la imprimantă se vor scrie valorile, aflate în memoria calculatorului, variabilelor din listă.

Exemplu:

5	6	7	WRITE (108, 111) I, VAL, Q, VK
111			FFORMAT(-----)

Interpretare: Scrie (WRITE) la imprimantă (108) conform indicațiilor date de instrucțiunea FFORMAT cu eticheta 111 valorile variabilelor: întregă I; reale VAL, Q și VK, valori aflate în memoria calculatorului.

4.3. Instrucțiunea **FORMAT**

Este o instrucțiune ajutătoare indicând modul cum trebuie citită sau scrisă o înregistrare. Ea poate fi plasată oriunde în program, de regulă însă se scrie imediat după instrucțiunea pe care o deservește.

Forma generală a instrucțiunii **FORMAT**:

et||**FORMAT**(listă)

în care: - et = etichetă, identică cu cea aflată în paranteza instrucțiunii de intrare/ieșire pe care o deservește instrucțiunea **FORMAT**;

- **FORMAT** = cuvânt cheie indicând funcționalitatea instrucțiunii;

- listă = o succesiune de coduri de format (numite și descriptori de zonă, descriptori de câmp sau specificații de format), despărțite prin virgulă și închise între paranteze.

Obs. 1. De regulă codurile sînt despărțite prin virgulă. Există și excepții care vor fi amintite la locul potrivit.

2. Dacă în lista de coduri nu apare codul / atunci lista se referă la o singură înregistrare, caz în care suma cîmpurilor codurilor nu trebuie să depășească 80 de coloane respectiv 132 de spații. Aceeași regulă trebuie aplicată pentru cazul apariției codului /, înșiruirii de coduri dintre paranteze stîngă și primul /, între slashuri și între ultimul / și paranteza de închidere a **FORMAT**-ului.

Codurile se împart în două categorii:

- a) - coduri de control al așezării datelor în înregistrări;
- b) - coduri pentru transmiterea datelor.

Codurile de control se vor numi cu un termen generic în cele ce urmează "coduri de paginare" deoarece se folosesc în

mod special, dar nu exclusiv, la realizarea unei anumite aşezări a rezultatelor în pagina imprimantăi.

4.3.1. CODURI DE PAGINARE

4.3.1.1. Codurile benzii pilot

Asigură deplasarea paginii de imprimantă într-o anumită poziţie şi anume (tabelul 4.1).

Tabel 4.1 Codurile benzii pilot

Cod	Efect
' (bland)	salt de rând
' 0 '	salt de două rânduri
' 1 '	pagină nouă (salt de pagină)
' + '	nu se deplasează (supraimprimare)
Orice alt caracter	salt de rând

Obs. Aceste coduri se scriu pe prima poziţie a listei de coduri ce se referă la scrierea unui rând la imprimantă.
Dacă ^{codul} nu este indicat în mod explicit, se consideră cod de control al imprimantăi primul caracter al rândului ce urmează a fi tipărit, care se va interpreta drept cod de salt de rând şi nu se va tipări ceea ce constituie o posibilă sursă de eroare (vezi exemplul dat la codurile pentru transmiterea datelor întregi).

4.3.1.2 Codul X

Forma generală:

nX

în care:

X - este codul de exceptare pe înregistrare;

n - constantă întreagă, diferită de zero, strict pozitivă, indicând numărul de coloane exceptate de la citire sau numărul de spații exceptate la scriere în raport cu cimpul codului precedent.

Exemplu - la citire:

4X arată că se vor excepta următoarele patru coloane de la citire, indiferent de conținutul acestora.

- la scriere:

10X arată că următoarele zece spații vor fi lăsate libere.

4.3.1.3. CODUL /

Codul bară înclinată sau slash este codul de sfârșit de înregistrare.

Regulă. Dacă n este numărul de / înserate la începutul sau sfârșitul listei de coduri se vor excepta n înregistrări; dacă cele $n > 1$ slash-uri sînt înserate în interiorul listei se vor excepta n-1 înregistrări; dacă n=1 se trece la înregistrarea următoare.

Exemplu Instrucțiunea

5|6|7

10|FORMAT (/// c₁, c₂, /// c₃, c₄, c₅ //)

se interpretează astfel:

- la intrare

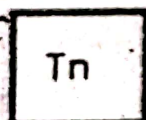
primale trei cartele se vor excepta de la citire, de pe a patra se vor citi valorile conform codurilor generice c_1, c_2 , următoarele trei cartele se exceptează de la citire (cartelele 5, 6 și 7), de pe cartela a opta se citesc valorile conform codurilor generice c_3, c_4, c_5 , iar ultimele două cartele se exceptează de la citire.

- la ieșire:

primale trei rânduri se exceptează de la scriere, pe al 4-lea se scriu datele conform codurilor generice c_1 și c_2 , următoarele trei rânduri sînt lăsate libere și pe următorul (al 8-lea) se vor scrie datele conform codurilor generice c_3, c_4, c_5 , iar următoarele două rânduri (al 9-lea și al 10-lea) sînt exceptate de la scriere.

4.3.1.4. CODUL T

Are forma generală:



unde: n - este constantă întregă, mai mare ca zero, fără semn, care arată că se exceptează $n-1$ elemente pe înregistrare începînd cu cel mai din stînga. Utilizarea codului T este avantajoasă la ieșire pentru afișarea unor coloane de tablou.

Exemplu: Instrucțiunea

51617
1 |

FORMAT (T 40, c_1) se interpretează:

- la intrare:

valoarea citită cu ajutorul codului generic c_1 se găsește începînd din coloana 40-a a cartelii de date;

- la ieșire:

valoarea ce urmează a fi tipărită cu ajutorul codului ge-

nerio c_1 se tipărește din a 40-a poziție a rîndului.

4.3.1.5 Codul H

Se folosește pentru transmiterea datelor Hollerith. Are forma generală:

$nH\text{text}$

în care: n = constantă întregă, fără semn, mai mare ca zero, indicînd numărul de caractere ce urmează în mod nemijlocit codului H în "text";

text = constantă de tip Hollerith.

Interpretare. Apariția în listă a unui cod H are ca urmare scrierea la imprimantă în poziția indicată pe rînd a mesajului cuprins în "text".

Exemplul. Secvența de program de mai jos are ca

5	6	7	
			WRITE (108, 10)
10			FORMAT ('0', 30 x, 'FAC.DE TEXTILE').

efect afișarea la imprimantă pe cel de-al 3-lea rînd (2 rînduri sînt lăsate albe datorită codului '0') după exceptare: a 30 de spații libere a textului.

FAC. DE TEXTILE

Fig.4.1. Macheta imprimantei pentru exemplul

de mai sus

Codul apostrof

Are forma generală:

'text'

Interpretare: Apariția în lista de coduri a primului apostrof indică existența unui mesaj, se caută cel de-al doilea apostrof de închidere și textul cuprins între cele două apostrofură se scrie la imprimantă în poziția respectivă.

Obs. Codurile H și apostrof sînt idempotente.

Se folosesc pentru introducerea în zona rezultatelor la imprimantă a unor texte cu caracter lămuritor, liniaturi, sublinieri, tabelări, curbe sau chiar figuri.

Exemplul: Să se întocmească un program pentru afișarea la imprimantă a unui tabel cu numele studenților unei grupe și adresa unde locuiesc conform modelului de mai jos:

Tabel cu stud.Gr.

Nr. crt.	Numele și Prenumele	Adresa
1.	Popescu V.	T ₄ -C 304
2.	Vasile P.	T ₄ -C 112
3.	Vlase I.	T ₃ -C 434
4.	Vicu I.	T ₄ -C 222

Secvența de program care asigură scrierea acestui tabel este următoarea:

```

5167
WRITE (108, 1)
1  FORMAT (45x, 'TABEL    CU STUD.GR.'/ 145x, 18(1H.))//
A 36x, 40(1H.)/ 36x, '. N R.CRT .  NUMELE SI PROM. .'
A  ADRESA  .'/ 36x, 37(1H.)/ 36x, ' . 1 .
A POPESCU V. . T4 - C 304 .'/ 36x, ' . .2
A  VASILE P. . T4 - C 112 .'/36x, ' .
A 3  . VLASE I. . T3 - C. 434 .'/
A 36x, ' . 4 . VIGU I. . T4 - C 222
A  .'/36x, (40(1H.))
    
```

Explicație: Conform indicațiilor date de instrucțiunea **FORMAT** cu eticheta 1 se va tipări pe un rând, după exceptarea de 44 de spații libere (primul din cele 45 a fost interpretat drept cod de salt de rând), textul cuprins între apostrofuri, titlul, se trece apoi pe rândul următor (/) pe care se realizează sublinierea titlului tabelului.

Această subliniere se realizează repetând de 18 ori codul 1 H = ceea ce are ca rezultat tipărirea unui șir de 18 asteriscuri. Se exceptează un rând (//), apoi se scrie linia superioară a tabelului utilizând în acest scop codul 1 H. repetat de 40 de ori, după care are loc tipărirea conținutului capului de tabel, care se realizează utilizând codul apostrof. După tipărirea pe rândul următor a liniei de închidere a capului de tabel

40(1H.)), se trece la tipărirea pe fiecare rând a unui nume de student și a adresei acestuia utilizând codul apostrof care va asigura scrierea conținutului a câte unui rând de tabel. La sfârșit se va scrie linia de închidere a tabelului prin același

cod 4a(1 E.). Rezultatul activității acestei secvențe de program va fi afișarea la imprimantă a unui tabel conform cu mărimea din exemplul problemei.

4.3.2. Coduri pentru transmiterea constantelor FORTRAN

4.3.2.1. Codul I pentru transmiterea constantelor întregi

Această formă generală:

rIn

în care:

r - factor de repetiție;

I - este codul pentru transmiterea datelor întregi (inițială de la cuvântul INTEGER);

n - câmpul codului, constantă întreagă, fără semn, diferită de zero, care indică numărul de coloane de pe care se face citirea sau numărul de spații în care se face scrierea unei valori întregi.

Deoarece codul poate fi utilizat într-un format de intrare sau într-unul de ieșire, se va discuta în cele ce urmează funcționalitatea acestuia în fiecare din cele două ipostaze.

La citire

În cele n coloane indicate ale câmpului se pot afla următoarele caractere:

- semnul plus (opțional) sau minus;
- cifre zecimale;
- coloane libere care se interpretează drept zero.

De regulă numerele se scriu în fiecare câmp, aliniate la dreapta. Apariția oricărui alt caracter FORTRAN în cele n coloane ale câmpului este interzisă și duce la întreruperea execuției programului.

Citirea se face începînd cu prima coloană, a cartelei de date, cea mai din stînga.

Exemplu: Fie instrucțiunea:

5	6	7	READ(105,1)I,K,M,NON,M2
1			FORMAT(I4,I5,I2,I4,I2)

└─ 1 2 5 7 4 2 4 1 0 ─┐ 3 3 3 4 ─┐

Obs.1. Cînd se discută instrucțiunea de intrare se are în vedere instrucțiunea propriu-zisă, FORMATUL aferent și cartela de date, așa cum se prezintă pe formularul de program PÖRTRAE.

Interpretare. Citește (READ) din setul de date (105) valorile variabilelor întregi I,K,M,NON,M2 conform indicațiilor date de instrucțiunea FORMAT cu eticheta 1, astfel: valoarea variabilei I se citește de pe primele 4 coloane ale cartelei de date conform codului I4, adică I=12, valoarea variabilei K se citește de pe următoarele 5 coloane (I5) adică K = 57424, valoarea variabilei M se citește de pe următoarele două coloane (I2), adică M=10; valoarea variabilei NON se citește de pe următoarele 4 coloane (I4), adică NON=333 și valoarea variabilei M2 de pe următoarele două coloane (I2), adică M2=40.

Obs.2. Cînd numerele conțin multe cifre zero se evită scrierea acestora (ca în cazul valorii variabilei M2), lăsînd coloana liberă (așa cum s-a mai arătat, coloanele libere sînt interpretate numai pe cartela de date drept zero).

La ieșire, pe cele n poziții indicate de cîmp, constanta apare scrisă în cele mai din dreapta poziții ale acestuia, spațiile neutilizate din partea stîngă rămînd libere, semnul -, dacă există, se scrie imediat înaintea celei mai

semnificative cifre. Dacă numărul de caractere necesare pentru scrierea valorii variabilei este mai mare decât cele n spații indicate de câmp, se vor afișa la imprimantă n asteriscuri. Această ultimă situație indică programatorului necesitatea măririi dimensiunii câmpului respectiv.

Exemplul: Fie instrucțiunea de scriere:

5	6	7
		WRITE (108,2)J1,K2,M2,I5
2		FORMAT (' ',10X,'J1=',I4,2X,'K2=',2x,
A		'M2=',I4,2x,'I5=',I4)

Interpretare. Scrie (WRITE) la imprimantă (108) valorile variabilelor J1,K2,M2,I5 conform indicațiilor date de instrucțiunea FORMAT cu etichetă 2, astfel: pe rând nou (' '), după 10 spații libere (10X), se scrie în clar ('J1=') valoarea variabilei J1 pe 4 spații (I4). După 2 spații libere (2x) se scrie în clar ('K2=') valoarea variabilei K2 pe 5 spații (I5) s.a.m.d.

Rezultatul activității acestei instrucțiuni este prezentat în macheta imprimantei din fig.4.2.

J1= 1265 K2= -724 M2=3500 I5=XXXX

Fig. 4.2 Macheta imprimantei

Obs.1. Ca și în cazul instrucțiunii de intrare când se discută instrucțiunea de ieșire se are în vedere instrucțiunea propriu-zisă, formatul aferent și macheta imprimantei.

Obs.2. Așa cum s-a arătat la momentul respectiv se recomandă afișarea în clar a rezultatelor pentru ușurarea înțelegerii acestora. Este evident că scrierea rezultatelor ca în fig.4.2 este net mai avantajoasă decât în fig.4.3, rezultat al activității instrucțiunii de mai jos.

Exemplu: Fie instrucțiunea:

5	6	7
WRITE (108,2)J1,K2,M2,I5		
2	FORMAT (10x,I4,2x,I5,2x,I4,2x,I4)	

1265	-724	3500	XXXXX
------	------	------	-------

Fig.4.3 Macheta imprimantei

Obs. Dacă un cod se repetă de mai multe ori în mod succesiv se va folosi un factor de repetiție corespunzător.

Așa, de exemplu, instrucțiunea

5	6	7
11	FORMAT(I4,I5,I5,I5,I3)	

este echivalentă cu instrucțiunea

5	6	7
11	FORMAT(I4,3I5,I3)	

4.3.2.2. Codurile pentru transmiterea datelor reale

Forma generală a acestor coduri:

rFn.l
rEn.l
rDn.l

în care F este codul pentru transmiterea datelor reale în forma CRB; E și D-coduri pentru transmiterea datelor reale în forma CRB cu exponent, prelucrate în simplă (E), respectiv dublă precizie (D);

r=factor de repetiție;

n=constantă întreagă, fără semn, diferită de zero, reprezentând lungimea câmpului pe suportul extern;

l=constantă întreagă, fără semn, ce reprezintă numărul de cifre de după punctul zecimal.

La intrare cele trei coduri au aceeași funcționalitate, adică, indiferent de forma de reprezentare pe suportul extern, reprezentarea internă este CRB.

Interpretarea: în cele m coloane indicate de cîmpul codu-
lui pot să apară:

- semnul plus (opțional) sau minus;
- cifrele zecimale: 0÷9;
- literele E sau D;
- punctul zecimal;
- spații libere, tratate drept zero;

Apariția oricărui alt caracter în cîmpul n este interzisă.

OBSERVAȚII - Dacă reprezentarea pe cartela de date este cu punct zecimal, specificația 1 din cod nu are efect, valoarea citită va avea numărul de cifre de după punctul zecimal, conform indicației cartelei de date.

- Dacă reprezentarea este fără punct zecimal, specificația 1 este operantă.

Codul F/E la intrare	Tabel 4.2.	
Valoarea pe suportul extern	Codul utilizat	Valoarea în memorie
342.72	F6.2/E6.2/D6.2	342.72
-96424.1	F8.3/E8.3/D8.2	-96424.1
5.3939E+02	F10.2/E10.2/D10.2	539.39
74.594E-01	F10.3/E10.3/D10.3	745.94
771114D+02	F8.2/E8.2/D8.2	771114.00
34542E03	F8.5/E8.5/D8.5	345.42000
34542E2	F7.1/E7.1/D7.1	345420.0
9425	F4.2	94.25
392524	F6.3	392.529
-1111111	F8.5	-11.11111
333	F3.3	333.0
76666632	F8.3	7 6666.632

Exemplu: Fie instrucțiunea:

	5	6	7
	READ(105,1)A,B,Q,T		
1	FORMAT(F4.1,F3.1,F2.2,F6.2)		
-7.1.	4	225-72147	

Din exemplu se observă că valoarea variabilei A va fi citită de pe primele patru coloane ale cartei de date (F4.1) cu o cifră după punctul zecimal, valoarea variabilei B va fi citită de pe următoarele 3 coloane (F3.1), dar poziția punctului de pe cartă nu coincide cu cea indicată de cod. În această situație B=.42, indicația codului fiind inoperantă. Valorile variabilelor Q și T citite cu codurile F2.2 și F6.2 sînt Q =.25, respectiv T = - 721.47. În aceste două cazuri indicația privind poziția punctului este operațională.

Obs. Este obligatorie respectarea condiției $n \geq 1$

La ieșire

Deoarece reprezentarea pe mediul extern al unei constante reale este diferită în forma F respectiv E sau D se va discuta utilizarea acestor coduri separat.

Utilizarea codului F la ieșire

Scrisura valorii se va face în cele mai din dreapta din cele n poziții indicate de câmpul codului. Dacă câmpul este prea mare, în pozițiile excedentare din stînga se înserează blancuri.

Dacă câmpul rezervat este prea mic, în cele n poziții se tipăresc tot atîtea asteriscuri.

Convertirea de la forma internă la cea externă se face cu rotunjire la 1 cifră zecimală.

Tabel 4.3

Variabile	Valoarea în memorie	Codul utilizat	Reprezentare externă
A	52.744	F6.3	52.744
B	-4.7277	F7.2	-4.73
C	12724.22	F6.2	12724.22
D	-9944211	F9.2	-99442.11

Utilizarea codurilor E și D la ieșire

Scrierea pe mediu extern se face în forma:

$S \ 0 . l \ e \ S_1 \ C$

în care: S - este semnul minus (-), (pentru semnul plus se inserează un blank);

l - mantisa

e - litera E sau D

S₁ - semnul exponentului;

C - exponentul format din două cifre zecimale, întregi.

Condiția necesară și suficientă pentru tipărirea valorii în forma de mai sus este ca:

$$n = 3 + 1 + 4 = 7 + 1$$

astfel:

- în primele trei poziții se tipărește semnul (dacă este minus), cifra zero (care reprezintă partea întreagă) și punctul zecimal;

- în următoarele 1 poziții se imprimă cele mai semnificative 1 cifre ale mantisei obținute prin trunchiere cu rotunjire;

- în următoarele 4 poziții se tipărește caracterul E/D, semnul +/- și două cifre zecimale;

- dacă $n > 1 + 7$ în stînga celor 1+7 caractere se

inserează blancuri...

Tabelul 4.4

Variabilă	Valoarea în memorie	Codul utilizat	Reprezentare externă
A	-0.245462	E11.5	-.24546E+00
B	72.324	E8.2	.72E+02
C	-0.00042	E12.5	-.42000E-03
D	-7524.3314924	D19.11	-.75243314924D+04
E	267831.41811	D11.8	.26783142D+06

Dacă se indică un cîmp n mai mic decît $l+7$, atunci forma de scriere se modifică astfel:

- dacă $n=l+6$ se renunță la spațiu pentru semn, dacă numărul este pozitiv și la cifra zero, reprezentînd partea întreagă, dacă numărul este negativ;

- dacă $n=l+5$ se renunță la poziția pentru semn dacă numărul este pozitiv și se tipăresc n asteriscuri dacă numărul este negativ;

- dacă $n=l+4$ sau mai puțin, pe imprimantă se vor tipări n asteriscuri (vezi tabelele 4.4 și 4.5)

Obs. Calculatorul FELIX-256 asigură scrierea în codurile E sau D fără zero la partea întreagă, deci în forma:

$S \cdot n \cdot S_1 \cdot C$

ca în exemplele variabilelor B, C, D, E de mai sus.

Utilizarea codurilor E și D la ieșire

Tabel 4.5

Variabilă	Valoare în memorie	Cod utilizat	Reprezentare externă
B	-75.12	E10.4	-.7512E+02
C	0.000424	E8.3	.424E-03
D	-92.999	E6.1	XXXXXXXX
E	0.094244	E9.6	XXXXXXXXXX

Pentru evitarea accidentelor prezentate în ultimele două exemple se recomandă a se utiliza așa numitele coduri acoperitoare, care au forma E 14.7 și D 21.14 evident funcție de precizia dorită a rezultatului.

Transmiterea datelor complexe

Pentru transmiterea datelor complexe se utilizează de două ori oricare din codurile pentru transferul datelor reale.

Obs. Datele complexe vor trebui declarate ca atare.

La intrare, valorile ce urmează a fi citite, părți componente a unei variabile complexe, apar ca două constante reale, una după alta.

Exemplu: Dacă variabila P are valoarea (35.42, -7.124) atunci pentru depunerea ei în memorie se folosesc instrucțiunile:

```

51617
|
| COMPLEX P
| READ (105,1) P
1 | FORMAT (F5.2,F5.3)
|
32.42 | - 7124

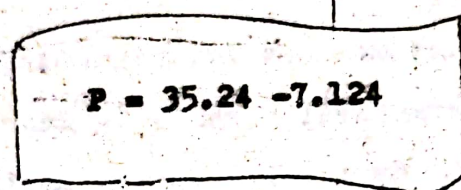
```


La ieşire. Partea reală şi partea imaginară ale unei variabile complexe se tipăresc ca două constante reale, una după alta.

Exemplu: Instrucţiunile:

5	6	7
		COMPLEX P
		WRITE (108,2)P
2		FORMAT (10x,'P-',F5.2,F7.3)

asigură tipărirea la imprimantă a valorii variabilei P, astfel



P = 35.24 -7.124

Fig. 4.4. Macheta imprimantă

4.3.2. 4. Codul L pentru transmiterea datelor logice

Are forma generală:

rLn

şi este folosit pentru introducerea şi extragerea datelor logice.

r = factor de repetiţie;

n = constantă de tip întreg, fără semn, diferită de zero, indicând numărul de coloane de pe care se face citirea, sau numărul de spaţii în care se face scrierea unei valori logice.

La intrare, în cele n coloane ale câmpului codului poate să apară orice şir de caractere FORTRAN. Primul caracter T sau F întâlnit, determină atribuirea valorii .TRUE. sau .FALSE. variabilei respective.

Dacă nu apare nici un caracter T sau F, valoarea atribuită variabilei logice este .FALSE.

Utilizarea codului L la intrare			Tabel 4.6.
Variabila logică	Reprezentarea externă	Cod utilizat	Valoarea în memorie
A	VATRA	L5	.TRUE.
B	bbFbTT	L6	.FALSE.
C	LULU	L4	.FALSE.
D	bbbbbb	L6	.FALSE.

La ieşire, în cea mai din dreapta poziţie din cale n indicate de cîmp, se scrie caracterul T sau F în funcţie de valoarea aflată în memorie, restul de $n - 1$ spaţii rămînînd libere.

Tabel 4.7			
Variabilă logică	Valoare în memorie	Cod utilizat	Reprezentare externă
A	.TRUE.	L2	bT
B	.FALSE.	L4	bbbf

4.3.2.5. Codul pentru transmiterea constantelor Hollerith

Codul A are forma generală:

$$rAn$$

se foloseşte pentru citirea sau scrierea constantelor Hollerith.

r = factor de repetiţie;

n = constantă întreagă fără semn, diferită de zero, indicînd mărimea cîmpului.

Obs. Se reaminteşte că utilizarea codurilor H şi apostrof pentru transmiterea şirurilor de caractere nu implică prelucrarea ca date FORTAN a acestora, în timp ce codul A indică transmiterea de valori unor variabile. Aceste variabile pot

fi de lungime $m=4$ sau $m=8$ octeți.

La intrare

Dacă $n \geq m$, se rețin ultimele m caractere din cele n ale șirului; dacă $n < m$ se rețin cele n caractere ale șirului aliniate la stînga restul de $m-n$ fiind blancuri.

Utilizarea codului A la intrare

Tabel 4.8.

Reprezentare externă	Codul folosit (n)	Lungimea variabilei (m)	Valoarea în memorie
BYTE	A4	4	BYTE
COD	A3	4	C0D0
TEXTIL	A6	4	XTIL
TESATURA	A8	8	TESATURA
TEXTILE	A8	8	TEXTILE0
TEHNOLOGIA	A 10	8	ENOLOGIA

Obs. Pentru evitarea segmentării datelor Hollerith se recomandă organizarea citirii acestora cu codul 20A4.

La ieșire are sens scrierea cu codul A dacă datele au fost memorate sub formă de caractere.

- dacă $n > m$, cele m caractere se scriu aliniate la dreapta în cele $n-m$ spații din stînga înscrinduse tot atîtea blancuri;
- dacă $n \leq m$, în cîmpul respectiv vor fi scrise numai primele n caractere din stînga șirului memorat.

Utilizarea codului A la ieșire

Tabel 4.9

Valoare internă	Lungimea variabilei	Cod utilizat	Reprezentare externă
DISC	4	A2	DI
DISC	4	A5	bDISC
CARTELA	8	A8	bCARTELA

4.4. Alte forme ale instrucțiunilor de intrare ieșire

4.4.1. Instrucțiunea de citire cu opțiuni

Forma generală a instrucțiunii READ este:

$[et] \parallel \text{READ}(n, et_1, \text{END}=n_1, \text{ERR}=n_2) \text{ listă}$

în care:

et - etichetă opțională numai dacă se fac referiri la această instrucțiune;

n - identificatorul perifericului folosit la introducerea datelor;

et_1 - eticheta instrucțiunii FORMAT.

n_1 = eticheta instrucțiunii ce se va executa la sfârșitul operației de citire a datelor;

n_2 = eticheta instrucțiunii ce se execută dacă apare o eroare ce nu a putut fi corectată de sistemul de operare

Parametrii $\text{END}=n_1$ și $\text{ERR}=n_2$ pot lipsi obținându-se forma standard a instrucțiunii de intrare sau se poate utiliza numai unul din ei.

Interpretare: CITEȘTE DE PE SETUL DE DATE (n) CONFORM INDICAȚIILOR DATE DE INSTRUCȚIUNEA FORMAT CU ETICHETA et_1 VALORILE VARIABILELOR DIN LISTA.

DACA S-A AJUNS LA SFÎRȘITUL SETULUI DE DATE TRECI LA INSTRUCȚIUNEA (n_1). DACA APARE O EROARE ÎN PROCESUL DE CITIRE TRECI LA INSTRUCȚIUNEA (n_2).

n_1 și n_2 sînt etichete de instrucțiuni executabile.

Parametrul $\text{END}=n_1$ se utilizează cînd programul prelucrează un număr variabil de constante. Pentru cititorul de cartele, cartela ce indică sfîrșitul de fișier (end of file) este :

1	2	3	4
.	E	O	F

4.4.2. Instrucțiuni intrare/ieșire FORTRAN II

Aceste instrucțiuni sînt recunoscute de compilatorul calculatorului FELIX C-256.

Ele au forma generală:

READ n, listă WRITE n, listă

cu semnificația cunoscută.

Ele se referă în mod implicit la cititorul de cartele și respectiv, imprimantă - casinguri periferice utilizate de calculator în perioada apariției versiunii II al limbajului FORTRAN.

Exemplu: Fie instrucțiunea:

5 6 7	
	READ 4, A, B
4	FORMAT (F4.1, F5.2)

Indică citirea de pe cartele conform instrucțiunii FORMAT cu eticheta 4 a valorilor variabilelor A, B.

Exemplu: Fie instrucțiunea:

5 6 7	
	WRITE 10, A, B
10	FORMAT (2x, F7.2, F9.3)

Indică scrierea la imprimantă conform indicațiilor date de FORMAT 10 valorilor variabilelor A și B.

Instrucțiunile:

PRINT n, listă PUNCH n, listă

în care: PRINT = cuvînt cheie cu semnul de TIPAREȘTE;

PUNCH = cuvînt cheie cu semnul de PERFORAȚIA.

Sînt echivalente cu instrucțiunea WRITE și se referă la imprimantă respectiv la perforatorul de cartele.

Exemplu: Instrucțiunea:

```
PRINT 10,A,B,C
10 FORMAT( 3x,3F10.4)
```

va efectua tipărirea la imprimantă a variabilelor A,B,C
conform cu indicațiile FORMATului.

10 Este echivalentă cu instrucțiunea:

```
10  WRITE 10,A,B,C
    FORMAT ( 3 I, 3 F10.4 )
```

4.4.3. Instrucțiuni de intrare/ieșire pentru tablouri:

Citirea sau scrierea unor tablouri ~~PORTRAN~~ se poate face folosind una din următoarele forme ale instrucțiunilor de intrare/ieșire:

- dacă se respectă regula implicită a indicilor atunci tabelul va apare ca o variabilă în listă, ca în exemplul de mai jos, iar instrucțiunea are forma cunoscută:

Exemplu:

```

5 6 7
|
| DIMENSION A(5,5)
| READ(105,10)A
10 | FORMAT(10F5.2)
    |
    |

```

In cazul acestui exemplu trebuie avut grijă ca elementele tabloului să se transcrie pe coloane pe cartelele de date(v.cap.3);

- dacă se dorește transferul datelor pe linie, lista de variabile are forma prezentată în cap.3, conform regulii explicite a indicilor, iar instrucțiunea de intrare/ieșire prezintă structura ca mai jos:

Exemplu: Fie secvența de program:

5 | 6 | 7

DIMENSION B(2,3)

READ(105,2) ((B(I,J),J=1,3),I=1,2)

2 | FORMAT (6F3.2)

În acest exemplu citirea datelor se va face pe linii (indicele coloanei variază cel mai repede), deci aşezarea valorilor pe cartela de date se va face în aceeaşi ordine.

Exemplu: Fie secvența de program:

5 | 6 | 7

DIMENSION x(3,3)

WRITE (108,10) ((X(I,J),J=1,3),I=1,3)

10 | FORMAT (3(30X,3(F3.1,3X)))

1.2	3.2	-.3
7.1	1.1	-.9
3.1	1.9	7.9

Fig. 4.5. Macheta imprimantei pentru exemplul de mai sus

Se observă că, în acest exemplu, tipărirea valorilor tabloului X s-a făcut în forma obişnuită deoarece programatorul a indicat în mod explicit modul cum trebuie să varieze indicii.

4.5. Relația dintre lista de intrare/ieșire și lista de

coduri FORMAT

Așa cum s-a arătat când s-a discutat forma generală a unei instrucțiuni de intrare/ieșire, lista enumeră variabilele implicate în transferul de valori, în timp ce lista de specificație **FORMAT** este o înșiruire de coduri sub controlul cărora se face

acest transfer.

Codurile X, T, H, apostrof și / nu au echivalent în lista de intrare/ ieșire.

Codurile I, F, E, D, L, A se folosesc pentru transmiterea datelor.

Controlul transferului de date este trecut pe rînd codurilor FORMAT în ordinea prezenței lor în lista de specificații de la stînga la dreapta.

Repetarea codurilor pentru transmiterea datelor se indică prin utilizarea unui factor de repetiție (constantă întreagă, fără semn, diferită de zero), în fața codului respectiv.

Astfel 3F4.2 înseamnă F4.2, F4.2, F4.2.

Grupe de coduri: un număr de coduri incluse între paranteze.

Repetarea unei grupe de coduri poate fi indicată prin factorul de repetiție.

De exemplu:

2(F4.2, I4) sau 2(F5.3, 2X)

Transmiterea datelor se face conform principiului: variabilele și codurile se corespund în ordinea prezenței lor în lista de intrare/ieșire respectiv în lista de specificații FORMAT și pe baza următoarelor reguli:

Regula 1: Lista de variabile, dacă există, trebuie satisfăcută în întregime.

Dacă numărul de coduri este mai mare decît numărul variabilelor, cele în exces sînt ignorate.

Dacă numărul de coduri este mai mic decît numărul de variabile; lista FORMAT se reexplorază, în vederea respectării regulii 1, după regulile:

Regula 2: Reexplorarea unei liste de coduri determină trecerea la o nouă înregistrare.

Regula 3: Reexplorarea se face de la începutul listei,

dacă lista nu prezintă grupe de coduri, sau de la factorul de repetiție a grupului de coduri care are paranteza de închidere cea mai apropiată de paranteza de închidere a listei **FORMAT**.

Exemplu: Fie instrucțiunea de citire:

5	6	7
		READ (105,1) F1,F2,A,B,I
1		FORMAT (2F2.0,F5.1,F4.2,I4,I6,I7)
3245-	1	3423524 12

În exemplul dat, variabilelor F1 și F2 li se atribuie valori citite cu ajutorul codului F2.0 repetat de 2 ori, respectiv 3245 și 45; variabilelor A,B și I li se atribuie valori citite cu ajutorul codurilor F5.1, F4.2, I4, respectiv valorile - 134.2, 35.24 și 12.

Codurilor I6,I7 nu le corespund variabile deci sînt ignorate.

Exemplu: Fie instrucțiunea:

5	6	7
		READ (105,10) A,B,C,D
10		FORMAT (F4.2,F5.1)
4424-	1	295
-33	7	719

Valorile A și B se citesc de pe prima cartelă, respectiv 44.24 și -129.5

Pentru variabilele C și D se reexplorază lista de coduri, citirea valorilor acestora făcîndu-se de pe a doua cartelă de date, respectiv valorile: -,33 și 771.9

4.6. Utilizarea virgulei la scrierea datelor pe cartelă

În cele discutate pînă acum, la întocmirea cartelei de date s-a ținut cont întotdeauna de indicația dată de cîmpul codului.

Atunci cînd sînt multe valori de scris pe cartelă, este dificil pentru programator să repete fiecare cîmp în parte.

Se recomandă o facilitate acordată de FORTRAN și anume transcrierea datelor pe cartelă, una după alta, despărțite prin virgulă, dar fără a lua în considerare la modul absolut mărimea cîmpului. Condiția ca citirea să fie corectă în această situație este ca în mod obligatoriu $n > m$ în care n este cîmpul codului iar m este numărul de coloane ocupate efectiv de valoare.

Exemplu: Să se citească valorile a 5 variabile reale de pe o cartelă pe care s-a respectat mărimea cîmpului (a) și de pe o cartelă în care s-a utilizat virgula (b).

Exemplul :

	<u>5</u>	<u>6</u>	<u>7</u>	
				READ 7,A,B,C,D,E
	<u>7</u>			FORMAT (F4.1,F4.3,F5.2,2F6.4)
a)	142	3	32	-754 4042 -314
b)	142,33	2,	-754,-4042,-314	

4.7. Alte instrucțiuni FORTRAN

4.7.1. Instrucțiunea STOP

Această instrucțiune indică sfîrșitul executării unui program și îi corespunde în schema logică blocul delimitator STOP.

Executarea instrucțiunii STOP nu înseamnă oprirea calculatorului, ci transferul controlului sistemului de operare în vederea introducerii unei lucrări (job) în execuție.

Intr-un program pot fi una sau mai multe instrucțiuni STOP. Poate fi etichetată și are forma generală:

[et]	STOP n
[et]	STOP

în care: et- este o etichetă definită într-o instrucțiune executabilă anterioară;

n - o constantă cu pînă la 5 cifre.

4.7.2. Instrucțiunea PAUSE

Are ca scop inserarea unor pauze în timpul rulării programului și, opțional, afișează la consolă un mesaj lămuritor pentru operator.

Forma generală:

[et]	PAUSE
[et]	PAUSE n
[et]	PAUSE 'mesaj'

în care: n - o constantă întreagă, fără semn, formată din maximum 5 cifre,

mesaj - un șir de caractere Hollerith în clar sau codificat;

et - etichetă opțională

Exemplu:

PAUSE 'MONTATI DISCUL 1'

4.7.3. Instrucțiunea END

Are sensul de "SFIRSIT" și indică compilatorului sfîrșitul programului sursă. Este obligatorie ca ultimă instrucțiune în orice program sau subprogram.

Are formă generală:

END

CAPITOLUL 5

INSTRUCȚIUNEA DE ATRIBUIRE

Instrucțiunile de atribuire au forma generală:

$$[et] \parallel v = e$$

în care: et- etichetă opțională

v - o variabilă de tip aritmetic sau logic;

e - expresie aritmetică sau logică.

Instrucțiunile de atribuire sînt de două feluri:

- instrucțiuni de atribuire aritmetică;
- instrucțiuni de atribuire logică.

5.1. Instrucțiunea de atribuire aritmetică

Are forma generală:

$$[et] \parallel v = e$$

eticheta et-fiind opțională

v - variabilă aritmetică neindexată sau indexată;

e - expresie aritmetică de orice tip.

Interpretare

Apariția în program a unei instrucțiuni de atribuire are drept urmare evaluarea expresiei aritmetice din membrul drept, convertirea rezultatului la tipul variabilei din membrul stîng, dacă este cazul, și atribuirea acestei valori drept valoare curentă variabilei din membrul stîng.

Exemple:

1	I=K+J
2	V=A*B/(C-D)
3	T=lnL/M

Considerând tipul variabilelor ca stare prin convenția FORTRAN predefinită în exemplele date, se observă că: în instrucțiunea 1 rezultatul din membrul drept este întreg, variabila din membrul stîng este întreagă, atribuirea este imediată. Același lucru se petrece și în cazul instrucțiunii 2.

În instrucțiunea 3 rezultatul evaluării membrului drept este întreg. Pentru a fi atribuit variabilei din membrul stîng, care este de tip real, trebuie să se convertească rezultatul la tipul real, apoi se face atribuirea.

Obs. Semnul = nu înseamnă egalitate ci atribuire.

Ca urmare variabila din membrul stîng poate să apară ca operand în membrul drept; situație în care valoarea variabilei din membrul drept se numește VALOARE TRECUTĂ a variabilei, iar valoarea atribuită se numește VALOARE ACTUALĂ SAU PREZENTĂ a variabilei respective.

Exemplu

A = B - CMA

Valoarea variabilei A din membrul drept este numită trecută deoarece ea trebuie să existe în momentul declanșării operației de evaluare a expresiei; valoarea atribuită variabilei A din membrul drept este actuală deoarece ea a apărut, acum, ca urmare a evaluării expresiei din membrul drept.

Pentru înțelegerea corectă a acestor noțiuni este bine să se revadă principiile de lucru ale calculatoarelor prezentate în capitolul 1.

5.2. Instrucțiunea de atribuire logică

Are forma generală:

$[et] || v = e$

în care: v - este o variabilă simplă sau indexată logică, declarată ca stare la începutul programului;

e - expresie de tip logic;

et - este o etichetă opțională.

Interpretare

Apariția în program a acestei instrucțiuni are drept urmare evaluarea expresiei logice din membrul drept și atribuirea valorii obținute drept valoare curentă variabilei din membrul stîng.

Exemplu

```

|| LOGICAL V, A, T, K
|| V = A.AND.NOT.K.OR.T.AND.(B.GT.O)
    
```


5.3. Instrucțiuni de inițializare

Instrucțiunea DATA

Are forma generală:

$DATA \ l_{x_1}/l_{v_1}/l_{x_2}/l_{v_2}/\dots/l_{x_n}/l_{v_n}/$

în care:

$l_{x_1}, \dots, l_{x_n}$ = liste de variabile, variabile indexate
sau nume de tablouri;

$l_{v_1}, \dots, l_{v_n}$ = liste de constante FORTRAN

Interpretare

Se atribuie drept valori inițiale variabilelor din listele $l_{x_1}, \dots, l_{x_n}$, valorile corespunzătoare din listele de constante $l_{v_1}, \dots, l_{v_n}$. Atribuirea se face în faza de compilare.

Folosirea instrucțiunii de inițializare se va face respectând următoarele reguli:

- dacă mai multe constante consecutive din listă au aceeași valoare se scrie valoarea o singură dată precedată de factorul de repetiție și de semnul $\times$; (de exemplu 4×0.1 înseamnă $0.1, 0.1, 0.1$);

- se va asigura obligatoriu corespondența biunivocă dintre numărul variabilelor din listele l_{x_1} și numărul constantelor din listele l_{v_1} ;

- dacă în instrucțiunea DATA se face referire la un tablou, atunci dimensiunea acestuia trebuie declarată anterior;

- instrucțiunea DATA nu precede instrucțiunea implicită de specificare. Poate apărea oriunde în program, dar niciodată într-o secvență de program a cărei activitate se reia.

Instrucțiunea DATA nu se poate aplica variabilelor dintr-un domeniu comun (v. cap. 9). În acest caz se va utiliza subprogramul BLOCK DATA.

Exemplu: Fie instrucțiunea;

DATA A,B,C,D,L/4.,5.2,2 $\times$ 3.3,T/

Această instrucțiune inițializează variabilele A și B cu valorile 4. și respectiv 5.2, variabilele C și D cu valorile 3.3 (apare factorul de repetiție 2 $\times$), iar variabila logică L (declarată ca stare) cu valoarea .TRUE.

Obs. Când se inițializează elementele unui tablou atenție la regula indicilor.

CAPITOLUL 6

INSTRUCȚIUNI DE SPECIFICARE

Instrucțiunile de specificare au rolul de :

- a compilatorului informații asupra naturii variabilelor folosite în programul sursă;
- a furniza informațiile necesare pentru a rezerva locuri în memorie pentru aceste variabile;
- a preceda orice apariție în program sau subprogram a variabilelor specificate, și de aceea ele se plasează la începutul programului sau subprogramului înainte de orice instrucțiune executabilă.

6.1. Instrucțiunea DIMENSION

Are forma generală:

DIMENSION $t_1(i_1), t_2(i_2), \dots, t_n(i_n)$

în care:

- DIMENSION - cuvânt cheie cu sensul de DIMENSIUNE;
- $t_1, \dots, t_n$ - nume de tablouri;
- $i_1, \dots, i_n$ - constante întregi, fără semn, în număr de 1 - 7, despărțite prin virgulă, reprezentând valorile maxime ale indicilor tabloului a cărui dimensiune se declară.

Pot fi variabile întregi numai în cazul unei proceduri.

Instrucțiunea DIMENSION are rolul de a informa calculatorul asupra dimensiunilor tablourilor în vederea rezervării spațiilor de memorie corespunzătoare. Se referă atât la tablouri cu date de intrare, cât și la cele cu date intermediare și finale.

Exemplu

DIMENSION TAB(100), L(2,5), B(5,7,7)

În exemplul de mai sus se declară dimensiunea tabloului TAB cu 100 de elemente, tabloul L cu 2x5, adică 10 elemente și tabloul B cu 5x7x7, adică 245 elemente.

6.2. Instrucțiuni de specificare a tipului

Există două moduri de a specifica tipul unei variabile sau a unui tablou în FORTRAN.

1. Specificarea implicită
2. Specificarea explicită

6.2.1. Instrucțiunea IMPLICIT

Are forma generală:

`IMPLICIT tip *i1(a1,a2,...),tip *i2(b1,b2,...)`

în care:

tip este unul din cuvintele cheie : INTEGER, REAL, COMPLEX, LOGICAL și DOUBLE PRECISION care specifică tipul variabilelor; i_1, i_2 - reprezintă lungimea asociată variabilelor (parametru opțional). Lipsa lui asigură atribuirea lungimii standard; $a_1, a_2, \dots; b_1, b_2, \dots$ - caractere alfabetic separate prin virgulă sau despărțite prin caracterul „-” reprezentând secvența de alfabet.

Instrucțiunile IMPLICIT au următoarele funcții:

- indică tipul unui grup de variabile sau tablouri prin caracterul inițial al numelui lor;
- indică mărimea domeniului de memorie care trebuie atribuit fiecărei variabile corespunzător tipului ei.

Interpretare

Apariția în program pe prima poziție, și în a doua poziție într-un subprogram, a unei instrucțiuni IMPLICIT are ca efect declararea ca fiind de tip-i- a tuturor variabilelor sau tabelor a căror nume simbolic începe cu una din literele ce urmează între paranteze.

Instrucțiunea anulează efectul convenției predefinite FORTRAN, care rămâne valabilă pentru grupul de litere nespecificate în instrucțiunea IMPLICIT.

Exemplu: Instrucțiunea:

`IMPLICIT INTEGER (A,B,D-G), REAL
(L-N,P), LOGICAL (Q,R)`

declară ca fiind întregi de lungime standard toate variabilele a căror nume simbolic începe cu litera A, B, D, E, F și G, ca fiind reale de lungime standard toate variabilele a căror nume începe cu una din literele L, M, N și P și ca fiind logice variabilele

a. căror nume simbolic încep cu una din literele Q și R.

Obs.

Intr-un program sau subprogram nu poate exista decât o singură instrucțiune IMPLICIT.

6.2.2. Instrucțiunea de specificare explicită

Are forma generală:

Tipul $V_1(k_1)/x_1, \dots, V_n(k_n)/x_n/$

în care:

tip - unul din cuvintele cheie: REAL, INTEGER, COMPLEX, LOGICAL;

n - parametru opțional indicând lungimea variabilelor; când lipsește se consideră lungimea standard.

$V_1 - V_n$ - listă de variabile sau tablouri cărora li se specifică explicit tipul;

$k_1 - k_n$ - indici numerici

$x_1 - x_n$ - valori de inițializare

Instrucțiunea de specificare explicită are rolul:

- de a indica tipul unei variabile sau tablou prin însăși numele simbolic;

- indică mărimea domeniului de memorie care trebuie atribuit fiecărei variabile, corespunzător tipului ei;

- indică dimensiunea tablourilor;

- atribuie valori inițiale pentru variabile și tablouri.

Instrucțiunea de specificare explicită este mai puternică decât instrucțiunea IMPLICIT deci și mai puternică decât convenția predefinită FORTRAN.

Exemplu:

```
REAL I, L, M(4,5)
LOGICAL T, LOG
INTEGER A(4,5)/ 3*4, 10*0, 7*3/
```

declară ca fiind de tip real variabilele I, L și tabloul M cu 4x5 elemente, ca fiind de tip logic variabilele T și LOG; ca fiind de tip întreg tabloul A cu 4x5 elemente, elemente care sînt inițializate cu valoarea 4 primele 3, cu valoarea 0 următoarele 10 și cu valoarea 3 ultimile 7, ordinea de atribuire fiind făcută în baza regulii predefinite a indicilor.

Aceste instrucțiuni sînt echivalente cu cele de mai jos:


```

DIMENSION M(4,5), A(4,5)
REAL I,L,M
LOGICAL T,LOG
INTEGER A
DATA A/3*4,1000,7*3/
    
```

6.2.3. Instrucțiunea DOUBLE PRECISION

Are forma generală:

DOUBLE PRECISION $v_1(k_1), v_2(k_2), \dots, v_n(k_n)$

în care:

- DOUBLE PRECISION - cuvânt cheie cu sensul de DUBLĂ PRECIZIE;
- $v_1, v_2, \dots, v_n$ - nume de variabile, tablouri sau de funcții;
- $k_1, \dots, k_n$ - liste de constante întregi fără semn, separate prin virgulă, în număr de 1 + 7 reprezentând valoarea maximă a fiecărui indice.

Instrucțiunea aceasta specifică în mod explicit că variabilele $v_1, \dots, v_n$ sînt de tip dublă precizie. Această instrucțiune invalidează orice specificație implicită sau prin convenția FORTRAN.

Este echivalentă cu instrucțiunea de specificare explicită REAL=8.

Nu poate defini valori inițiale.

Exemplu:

```

DOUBLE PRECISION A,B(2,10)
    
```

Instrucțiunea declară în dublă precizie variabila A și elementele tabloului B.

Este echivalentă cu secvența de program:

```

|| DIMENSION B(2,10)
|| DOUBLE PRECISION A,B
    
```


CAPITOLUL 7

INSTRUCȚIUNI DE SALT

Aceste instrucțiuni au rolul de a întrerupe desfășurarea normală secvențială a programului (executarea succesivă a instrucțiunilor programului) și reluarea activității acestuia de la o anumită instrucțiune identificată printr-o etichetă, în funcție de logica programului.

7.1. Instrucțiunea de salt necondiționat

7.1.1. Instrucțiunea GØ TØ

Are forma generală:

[et] GØ TØ n

în care: - GØ TØ - cuvânt cheie cu sensul de "transfer la";
 - n - etichetă a unei instrucțiuni executabile de la care se reia desfășurarea programului;
 - et - etichetă opțională.

Interpretare: apariția în program a acestei instrucțiuni are ca urmare: **RELUAREA ACTIVITĂȚII PROGRAMULUI DE LA INSTRUCȚIUNEA CU ETICHETA N.**

Obs. Instrucțiunea ce urmează instrucțiunii GØTØ trebuie neapărat etichetată, în caz contrar nu se va executa niciodată.

Exemplu:

	A=BxC.
	A1 = A + D
	GØ TØ 4
1	x=x+y
	y=x/5
4	WRITE(108,10)A,B

Din secvența de mai sus se observă că după calculul variabilei A1 se trece direct la scrierea valorilor A,B ignorându-se

instrucțiunile următoare.

7.1.2. Instrucțiunea GOTO calculat

Are forma generală:

[et]|| GOTO($n_1, n_2, \dots, n_m$), I

în care: $n_1, \dots, n_m$ - etichete ale unor instrucțiuni executabile;

I - variabilă întreagă care poate lua valori de la

1 la m.

Dacă e necesar, instrucțiunea poate fi etichetată.

Apariția în program a acestei instrucțiuni are ca urmare transferul executării programului la instrucțiunea cu eticheta n_1 dacă $I=1$, la instrucțiunea cu eticheta n_2 dacă $I=2$ ș.a.m.d. Valoarea variabilei I trebuie definită înainte de instrucțiunea GOTO respectivă.

Exemplu:

		I=K+1
		GOTO(5,10,15,20),I
5		A=A/B
10		A=A-B
15		A=B
20		A=SQRT(B)

Se observă, din secvența de program prezentată mai sus, că I va lua valori în urma executării instrucțiunii de atribuire ce precede instrucțiunea GOTO. Astfel dacă $I=2$ se reia activitatea programului de la instrucțiunea ce are eticheta 10. Dacă I va lua valori în afara intervalului 1-4 instrucțiunea GOTO respectivă este ignorată.

7.1.3. Instrucțiunea GOTO impus

Are formă generală:

		ASSIGN $n, T0$ I
[et]		GOTO I, ($n_1, n_2, \dots, n_m$)

în care: - et este eticheta opțională;

- $n_1, n_2, \dots, n_m$ - etichetă de instrucțiuni executabile;

- I = variabilă întreagă care se definește prin

instrucțiunea de asignare ce precede obligatoriu instrucțiunea GOTO.

Apariția în program a acestei instrucțiuni GOTO asociat are drept urmare trecerea la executarea instrucțiunii cu eticheta n_1 indicată de instrucțiunea ASSIGN.

Exemplu:

10	ASSIGN 3 TO I
3	GOTO I(3,4,5)
	ALFA=ALFA*3.14/180
	ASSIGN 4 TO I
	GOTO 10

7.2. Instrucțiunea de salt condiționat

7.2.1. Instrucțiunea IF aritmetic

Are forma generală:

$[et] \parallel IF(e) n_1, n_2, n_3$

în care: IF - cuvânt - cheie cu semnul "dacă";

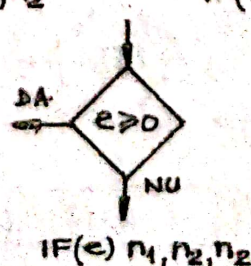
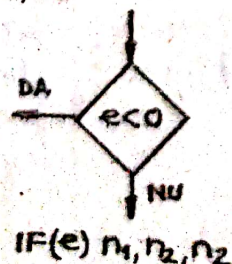
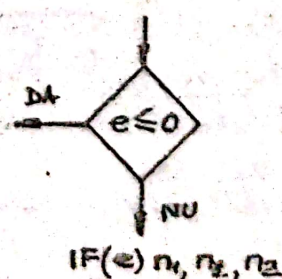
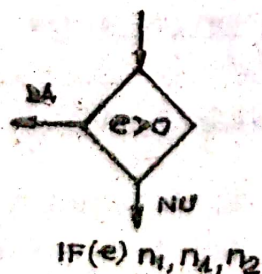
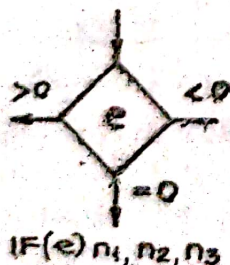
e - expresie aritmetică;

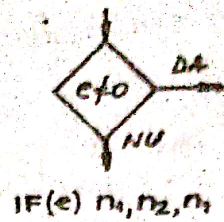
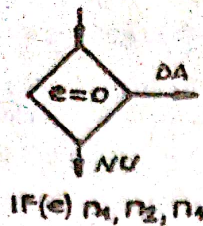
n_1, n_2, n_3 - etichete ale unor instrucțiuni executabile;

et - etichetă opțională.

Interpretare: DACA VALOAREA LUI $e < 0$ treci la INSTRUCȚIUNEA CU ETICHETA n_1 , DACA $e = 0$ TRECİ LA INSTRUCȚIUNEA CU ETICHETA n_2 , DACA $e > 0$ TRECİ LA INSTRUCȚIUNEA CU ETICHETA n_3 .

Instrucțiunea IF traduce în limbajul FORTRAN blocul logic din schema logică în următoarele variante :





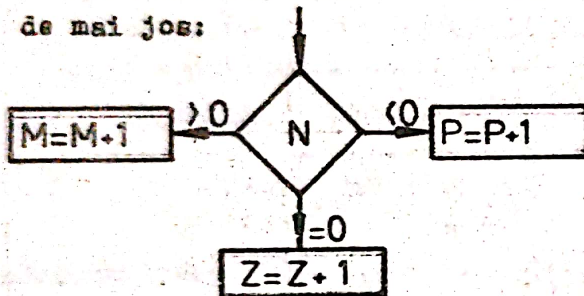
Se observă că, în funcție de condiția impusă, cele 3 etichete sînt distincte sau două din ele identice.

Obs. 1. Instrucțiunea ce urmează în mod nemijlocit instrucțiunii IF trebuie etichetată, în caz contrar nu poate fi niciodată executată.

2. Instrucțiunea IF poate fi etichetată.

Exemplu:

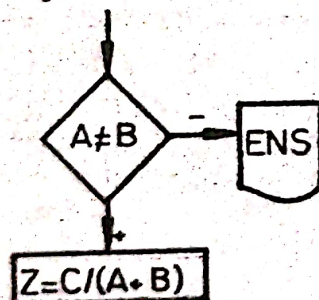
Fie secvența de schemă logică și secvența de program aferentă de mai jos:



	F(N) 10, 20, 30
10	M=M+1
20	Z=Z+1
30	P=P+1

Exemplu:

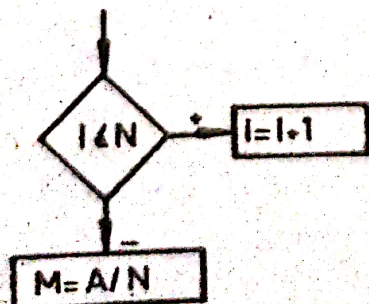
Fie secvența de schemă logică și secvența de program aferentă de mai jos:



	IF(A-B) 1, 2, 1
1	Z=C/(A-B)
2	WRITE 3
3	FORMAT(10, 'ENAS')

Exemplu:

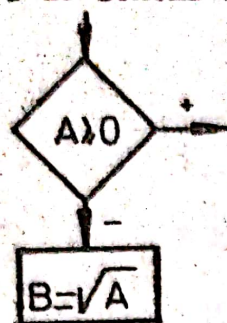
Fie secvența de schemă logică și secvența de program aferentă de mai jos:



	IF(I-N) 3, 3, 4
3	I=I+1
4	M=A/N

Exemplu:

Fie secvența de schemă logică și secvența de program aferentă de mai jos:



1	IF(A) 1,2,2
2	GOTO 4
	B = SQRT(A)

În cazul exemplului a) secvența de program traduce secvența din schema logică alăturată, care se interpretează astfel: dacă $N < 0$ se reia activitatea programului de la instrucțiunea cu eticheta 10, dacă $N = 0$ se reia activitatea programului de la instrucțiunea cu eticheta 20 iar dacă $N > 0$ se reia activitatea programului de la instrucțiunea cu eticheta 30.

În continuare, desfășurarea programului este normal-secvențială ca pînă în momentul apariției instrucțiunii IF.

În exemplele b ÷ d sînt prezentate trei secvențe de scheme logice cu blocuri de decizie în care se folosesc operatorii de ordonare ($\neq, \leq$ și $\geq$). Corespunzător acestor situații în secvențele de program întocmite apar instrucțiuni IF la care două din cele trei etichete sînt identice. Sintactica instrucțiunii are aceeași formă ca și în cazul precedent.

7.2.2. Instrucțiunea IF logic

Forma generală:

$[et] \parallel IF(e) a$

în care: e - expresie logică;

a - instrucțiune executabilă, alta decît IF logic sau DØ.

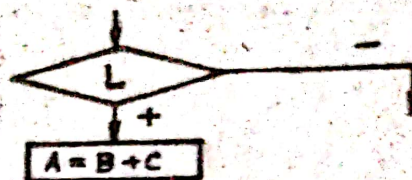
Interpretare : Dacă valoarea expresiei logice e este .TRUE. se execută instrucțiunea a, dacă valoarea expresiei logice e este .FALSE. se trece la executarea instrucțiunii imediat următoare instrucțiunii IF.

Obs. Dacă a este o instrucțiune de salt necondiționat (cel mai frecvent caz), are loc deci saltul indicat.

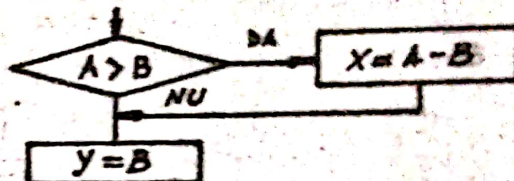
Dacă a nu este instrucțiunea de control, atunci după executarea acesteia se execută instrucțiunea imediat următoare instrucțiunii IF, situație exemplificată în cele două secvențe care urmează mai jos.

Exemplu:

a) IF(L) GOTO 4
A=B+C



b) IF(A.GT.B) X=A-B
Y=B-A



In cazul a) pentru $L = .TRUE.$ se execută instrucțiunea GOTO 4, iar pentru $L = .FALSE.$ se execută instrucțiunea $A=B+C$. In cazul b) dacă $A > B$ se execută instrucțiunea $X=A-B$, apoi instrucțiunea $Y=B$. Dacă $A \leq B$ se execută instrucțiunea $Y=B-A$.

CAPITOLUL 8

INSTRUCȚIUNEA DE CICLARE

În practica prăbușării datelor pe calculator sînt frecvente cazurile cînd un același calcul sau grup de calcule sînt realizate de mai multe ori, de fiecare dată înșă, pentru noi valori. Această situație este prezentă în schema logică prin cicluri, iar în limbajul FORTRAN se traduce prin utilizarea instrucțiunii DØ. Forma generală a instrucțiunii DØ :

$$[e] \parallel DØ \text{ } n \text{ } i = m_1, m_2, m_3$$

în care: *et* - etichetă ce poate să lipsească;

DØ - cuvînt rezervat cu sensul "EXECUTA";

n - etichetă a unei instrucțiuni executabile ce constituie linia finală a domeniului DØ.

Domeniul DØ- grupul instrucțiunilor executabile cuprinse între instrucțiunea DØ și instrucțiunea cu etichetă *n* inclusiv.

i = variabilă neindexată de tip întreg numită contor, numărător de pași sau variabilă de ciclare.

*m*₁, *m*₂, *m*₃ - constante întregi, fără semn, diferite de zero sau variabile întregi ce trebuie definite în momentul apariției în program a instrucțiunii DØ ce le folosește, reprezentînd:

*m*₁ - valoarea inițială a contorului;

*m*₂ - valoarea finală a contorului;

*m*₃ - pasul contorului (rația). Dacă *m*₃ lipsește din linia de definiție, se consideră 1.

Obs. Linia finală a ciclului DØ este orice instrucțiune executabilă, cu excepția acelor care prin natura lor împiedică continuarea operației de ciclare (alt DØ, IF de orice tip, GOTO de orice tip, STOP, PAUSE, RETURN).

Interpretarea instrucțiunii
 EXECUTA INSTRUCTIUNILE CARE URMEAZA PINA LA INSTRUCTIUNEA CU
 ETICHETA m INCLUSIV PENTRU VALORILE CONTORULUI i CARE VARIAZA DE LA
 m_1 PINA LA m_2 CU PASUL m_3 .
 Schema logică a acestei instrucțiuni are forma din fig.8.1

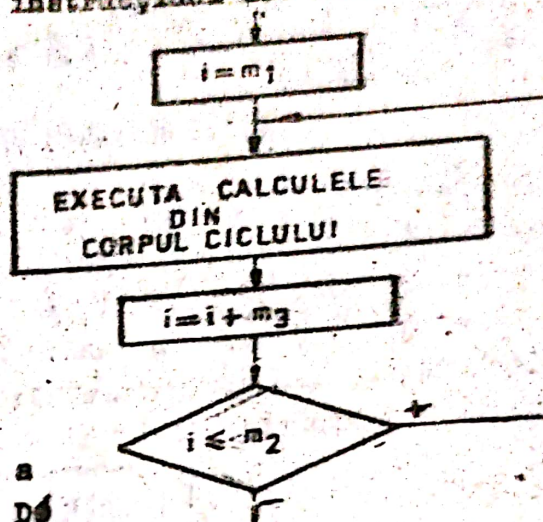


Fig.8.1 Schema logică a
 instrucțiunii D0

Apariția în program a unei instrucțiuni D0 are drept urmare:

- întreruperea desfășurării normal secvențiale a pro-
gramului;
- inițializarea variabilei i cu valoarea m_1 ($i = m_1$);
- executarea instrucțiunilor din corpul ciclului;
- creșterea cu rația m_3 a valorii variabilei de cicla-
re i ($i = m_1 + m_3$);

- compararea noii valori a lui i cu valoarea finală m_2 .

Dacă $i \leq m_2$ se reia activitatea de calcul asupra grupului
 respectiv cu noua valoare a lui i ; dacă $i > m_2$ se întrerupe ope-
 rația de ciclare, îndeplinindu-se condiția ieșirii din ciclu, și
 se transferă controlul executării programului la instrucțiunea
 imediat următoare instrucțiunii finale a ciclului (instrucțiunea
 cu eticheta m).

Instrucțiunea D0 determină executarea de $\left(\frac{m_2 - m_1}{m_3} + 1 \right)$
 ori a domeniului D0. S-a notat prin $\left[\frac{m_2 - m_1}{m_3} \right]$ partea întreagă a
 cîtelui $\frac{m_2 - m_1}{m_3}$, adică cel mai mare număr întreg care nu de-
 pășește pe $\frac{m_2 - m_1}{m_3}$.

Exemplu:

Secvența de program de mai jos:


```

2 | DØ 2 I=1,70,4
  | X(I) = A(I) + B(I)
  | AL=AL * SQRT(BL)

```

se interpretează astfel: se va executa calculul valorilor variabilelor $X(I)$, începînd cu $X(1)$ continuînd cu $X(5)=X(1+4)$ pînă cînd valoarea variabilei de ciclare I mărită de fiecare dată cu rația 4 depășește valoarea finală 70.

În momentul respectiv se întrerupe ciclarea și se execută instrucțiunea ce urmează imediat instrucțiunii cu eticheta 2.

Exemple de instrucțiuni DØ definite greșit:

DØ 4 K=1,72,-4	rația este o constantă întreagă negativă
DØ A J=2,24	eticheta nu este corect definită
DO 7 I=1,33,3	DO este un nume de variabilă și nu comanda DØ
DØ 5 V=7,77,7	variabila de ciclare trebuie să fie de tip întreg
DØ 3 M=0,99	parametrii de ciclare sînt constante întregi diferite de zero
DØ 44 L=0.5,77.3	parametrii de ciclare sînt constante de tip <u>întreg</u>

Reguli pentru utilizarea instrucțiunii DØ

1. Variabila de ciclare I poate fi utilizată de mai multe cicluri DØ în aceeași unitate de program, dacă acestea sînt consecutive.

Exemplu corect

```

4 | DØ 4 I=1,44,3
  | A(1)=B(1) * AL(i)
  | DØ 5 I=2,794,7
5 | V(1) = Q(1)-T(1)

```

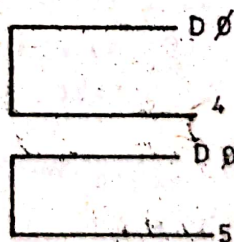


Fig.nr. 8.2

și exemplu incorect

```

7 | DØ 7 K=7,777,7
  | DØ 6 K=1,99,2
  | V(K)=I(K)-B(K)
6 | B1(K) = A(K)+ B2(K)

```

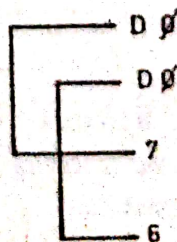


Fig.nr. 8.3

deoarece variabila K , definită cu valoarea 7 de linia $D\emptyset$ 7, își va modifica la următorul ciclu $D\emptyset$ 6 valoarea în mod artificial, nu conform definiției instrucțiunii $D\emptyset$.

2. Două sau mai multe cicluri $D\emptyset$ se pot cuprinde unul în altul. Primul se numește $D\emptyset$ exterior, celălalt $D\emptyset$ interior. Este obligatoriu ca toate instrucțiunile ciclului $D\emptyset$ interior să fie cuprinse în ciclul $D\emptyset$ exterior.

Exemplu corect:

	$D\emptyset$ 4	$I=1,70$
	$D\emptyset$ 5	$y=3,99,4$
5		$B(y)=B(y)/A(i)$
4		$T(i)=T(i)+V(i)$

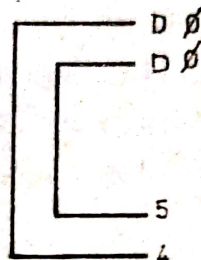


Fig.nr. 8.4

Exemplu incorect:

	$D\emptyset$ 5	$K=7,94,3$
	$D\emptyset$ 6	$M=10,100,2$
5		$D(K) = B(K)-A(M)$
6		$AL(M)=-BL(M)$

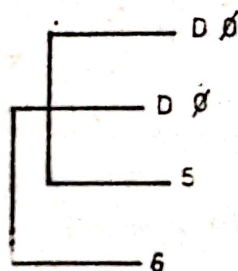


Fig.nr.8.5

deoarece cele două cicluri se întretaie.

3. Instrucțiunea finală poate să coincidă pentru două sau mai multe cicluri $D\emptyset$, ca aparținând fiecăruia dintre cicluri, dar nu poate fi referită decât din cel mai interior $D\emptyset$.

Exemplu

	$D\emptyset$ 7	$I=1,10$
	$D\emptyset$ 7	$J=1,7$
7		$C(I,J)=A(I,J)+B(I,J)$

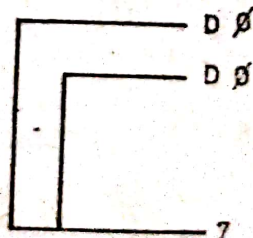


Fig.nr. 8.6

5. Transferul dintr-un domeniu $D\emptyset$ este permis, caz în care variabila de ciclare rămâne definită cu valoarea avută în momentul efectuării transferului.

Exemplu
DØ 11 I=3,94,4
A(1)=B(1)-AL(1)
L=I-2
IF(L)=10,10,11
11 C(1)=V(1)/A(1)

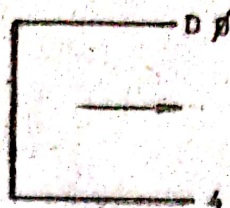


Fig.nr. 8.7

Se observă că pentru valori negative sau egale cu zero ale variabilei L se execută un transfer în afara ciclului.

6. Transferul din exterior într-un domeniu DØ este permis numai dacă e e s t aceste urmare a unui transfer în exterior din același domeniu DØ, iar parametri de ciclare nu au fost modificați în timpul transferului respectiv în domeniul exterior. (Instrucțiunile din exteriorul ciclului la care se face transferul constituie domeniul exterior al ciclului DØ).

Exemplu
DØ 10 I=2,75
DØ 11 J=1,34,3
V=A/J
IF(V) 12,12,22
22 V₁=SQRT(V)
100 B=A+V
11 D(J)=D(J)/B
10 L(1)=L(1)+V
12 V=ABS(V)
GOTO 100

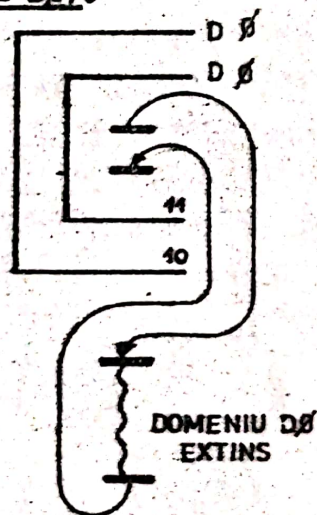


Fig.nr. 8.8

Din exemplu se observă că pentru valoarea mai mică sau egală cu zero a variabilei V se execută un salt în exteriorul ciclului DØ 11, apoi, după execuția instrucțiunii 12, se execută un salt (GOTO) în interiorul aceluiași DØ 11.

Nu sînt permise transferuri din exterior, reprezentate de schema din figura nr. 8.9.

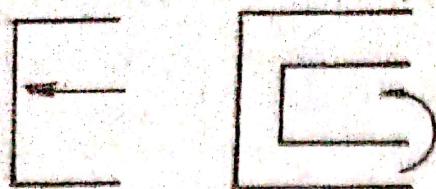


Fig.nr. 8.9

7. Valorile variabilei de ciclare, cît și a parametrilor de ciclare, pot fi modificați în afara domeniului D_0 cu condiția ca să nu se mai revină în ciclul respectiv.

Instrucțiunea CØNTINUE

Forma generală:

$e \parallel \text{CØNTINUE}$

Este o instrucțiune executabilă . Poate apare oriunde în program fără să influențeze ordinea normală de executare a instrucțiunilor.

Se folosește ca ultimă instrucțiune din ciclul D_0 atunci cînd există situația (din logica programului) ca ciclul să se termine într-una din instrucțiunile interzise.

Exemplu

	D_0	7	$I=1,77$
			$A=B(I)/M$
7			CØNTINUE
1			IF(I) 30,30,10

În exemplul de mai sus s-a evitat terminarea ciclului prin instrucțiunea IF prin montarea înaintea acesteia a instrucțiunii CØNTINUE cu eticheta 7, care permite realizarea integrală a ciclării.

Capitolul 9 PROCEDURI FORTRAN

9.1. Generalități

În rezolvarea unor probleme apar frecvente cazuri când un anumit calcul sau grup de calcule se efectuează de mai multe ori, de fiecare dată, cu alte date. Programul, care ar rezolva o astfel de problemă, întocmit după cunoștințele de până acum, ar avea o lungime mare, ceea ce ar fi neeconomic, atât sub aspectul consumului muncii programatorului, cât și a timpului de rezolvare pe calculator. Pe de altă parte, o serie de probleme fie că cer cunoștințe de matematică superioară care nu sînt la îndemîna multor programatori, fie că cer efectuarea unor calcule simple, de rutină, dar care aplicate unei mulțimi de date, de cele mai multe ori duc la erori. Firmele producătoare de calculatoare livrează pachete de programe prefabricate numite proceduri, care rezolvă o serie de astfel de probleme sau, uneori, însuși utilizatorul își organizează astfel de pachete de programe privind rezolvarea unor probleme specifice domeniului, grupate în biblioteci. Astfel de proceduri pot fi organizate și pentru uzul unui singur program.

Procedură (subprogram) - descrierea în limbajul FORTRAN a unui algoritm de interes general.

Procedura este definită în momentul descrierii în FORTRAN folosind parametrii formali. Referirea procedurii se face prin numele procedurii însoțită de parametrii actuali.

Limbajul FORTRAN folosește cinci tipuri de proceduri:

- funcții definite
- funcții intrinseci
- funcții externe uzuale
- proceduri FUNCTION
- proceduri subrutine

9.2. Funcții definite

În diferite tratate se mai întîlnesc sub denumirea de

Definirea funcției: formulă se realizează printr-o instrucțiune neexecutabilă ce trebuie plasată în program, înaintea oricărei instrucțiuni executabile, cu forma generală:

$$i(f_1, f_2, \dots, f_n) = e$$

Referirea la funcția internă se face într-o instrucțiune de atribuire unde numele funcției, însoțit între paranteze de parametrii actuali, apare ca operand.

51617.

```

-----
-----
F(X,Y) = X ** 3 / Y + X * Y - X * SQRT(Y)
-----
-----
-----
8 T1 = A + B / F (A,B)
-----
-----
9 WAL = T + E - S F (X1,Y1)

```

În secvența de program de mai sus se face definirea funcției interne F cu parametrii formali X și Y . În momentul când, în program apare instrucțiunea executabilă cu eticheta 8, se va evalua în primul rând funcția internă F prin înlocuirea parametrelor.

trilor formali X și Y în expresia din membrul drept, cu parametri actuali A și B, iar rezultatul evaluării înlocuiește în instrucțiunea cu eticheta B operandul P (A,B).

În același mod se procedează și în cazul instrucțiunii cu eticheta 9.

Observații

- 1) parametri formali pot fi utilizați în definirea mai multor funcții interne;
- 2) parametri actuali trebuie să concorde cu parametri formali ca tip, număr și ordine;
- 3) Parametrii actuali nu pot fi:
 - funcția însăși
 - o expresie ce conține funcția însăși
 - o funcție în care se folosește drept parametru funcția respectivă.
- 4) Numele funcției nu poate să apară într-o instrucțiune **EXTERNAL**;
- 5) Numele funcției nu trebuie să apară ca nume de variabilă în același program.

9.3 Funcții intrinseci

Sînt funcții utilizate foarte frecvent, din care motiv sînt incluse în bibliotecă și sînt emulabile. Ele sînt predefinite pentru compilator, fiind înscrise în programul apelant de către editorul de legături. Lista lor este prezentată în tabelul 3.6.

Apelarea se face folosind, drept operand în membrul drept al unei instrucțiuni de atribuire, numele funcției, însoțit între paranteze de parametri actuali, separați prin virgulă.

Observații

- numele funcției intrinseci este cuvînt rezervat, adică nu poate să apară ca identificator de variabilă sau de funcție internă;
 - nu poate să apară într-o instrucțiune **EXTERNAL**.
- De exemplu: **MAX (X,Y)**, **ABS (X)**, **FL0AT (M)**.

9.4. Funcții externe uzuale

Funcțiile externe uzuale sînt subprograme, prefabricate, livrate beneficiarului de către producătorul calculatorului.

Tabelul funcțiilor externe uzuale din biblioteca calcula-

torului FELIX C-256 este dat mai jos (tabelul 9.2).

Apelarea acestor funcții se face în același mod ca și în cazul funcțiilor intrinseci, descrise mai sus.

De exemplu: SIN (X), COS (X), LOG 10 (X).

9.5 Proceduri ... FUNCTION

Sînt unități de program independente numite proceduri sau subprograme. Aceste programe livrează programului principal cel puțin o singură valoare, așa cum, de altfel, se întîmplă și la utilizarea procedurilor discutate anterior.

Pot fi utilizate de mai mulți programatori și pot fi introduse într-o bibliotecă de subprograme.

Structura generală a unei proceduri FUNCTION este următoarea:

```

[tip] FUNCTION nume(a1, a2, ..., an)
[instrucțiuni de specificare]
Instrucțiuni executabile (cel puțin una)
RETURN
END
    
```

Linia de definiție cuprinde:

- opțional-tipul funcției;
- cuvîntul cheie FUNCTION care arată tipul subprogramului;
- nume - numele subprogramului;
- a₁, a₂, ..., a_n - parametrii formali (cel puțin unul). Acești parametri formali pot fi variabile simple, tablouri, variabile indexate, nume de subprograme FUNCTION sau SUBROUTINE deja definite.

Parametrii formali nu pot apărea într-o instrucțiune EQUIVALENCE, COMMON și DATA.

Declarațiile de tip sau dimensiune în corpul subprogramului sînt opționale.

Instrucțiunile executabile sînt obligatorii, din care una trebuie să fie de forma

nume = expresie

în care; nume este numele funcției neînsoțit de parametri ;

expresie - expresie aritmetică sau logică al cărei rezultat se

atribuie funcției.

Corpul subprogramului FUNCTION trebuie să conțină cel puțin o instrucțiune RETURN, care asigură revenirea în programul apelant. Execuția instrucțiunii RETURN asigură transmiterea valorii din subprogramul FUNCTION în locul de unde a venit apelul.

Subprogramul FUNCTION nu poate cuprinde instrucțiunile BLOCK, DATA, FUNCTION sau SUBROUTINE.

Instrucțiunea END indică compilatorului sfârșitul programului.

Apelarea subprogramului FUNCTION se face prin scrierea numelui subprogramului, însoțit între paranteze de parametrii actuali, drept operand într-o instrucțiune de atribuire.

În momentul apelării calculatorul caută subprogramul FUNCTION cu numele respectiv, înlocuiește parametrii formali cu cei actuali și rezolvă subprogramul. Revenirea în programul principal are loc, așa cum se arată mai sus, prin înlocuirea operandului din instrucțiunea apelantă cu valoarea transmisă de subprogram (v.fig.9.1)

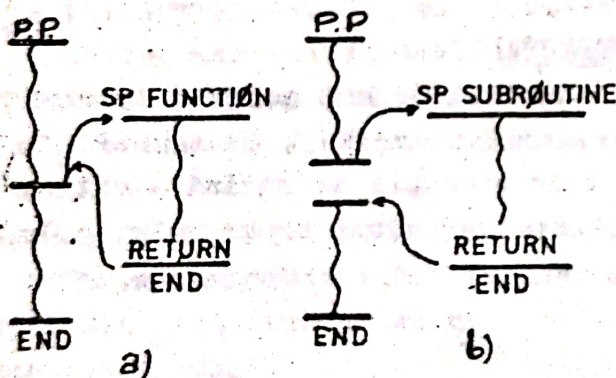


Fig. 9.1. Schema utilizării subprogramului
FUNCTION și SUBROUTINE

Parametrii actuali trebuie să coincidă în număr, tip și ordine cu parametrii formali.

Intrarea într-un subprogram FUNCTION se face fie prin linia

de definiție fie prin punctul definit de 4 instrucțiune ENTRY
(v. scap 9.8)

Exemplu

Procedură SUMA

```
FUNCTION SUMA (X,N)
DIMENSION X (N)
SUMA = 0
DO 1 I = 1,N
1 SUMA = SUMA + X (I)
RETURN
END
```

Program apelant

```
4 YN = SUMA (Y,N)/N
```

Procedura de tip FUNCTION cu numele SUMA, cu parametrii formali X,N, rezolvă suma unui șir de valori. Se observă că în corpul procedurii apare instrucțiunea 1, în care numele procedurii apare neînsoțit de parametru, iar în membrul drept-o expresie aritmetică.

În programul principal apare în instrucțiunea 4, drept operand, SUMA (Y,N). În acest moment se apelează procedura SUMA, care va calcula suma șirului Y citit în programul principal, anterior apelării, iar rezultatul se introduce în instrucțiunea apelantă.

9.6 Procedura SUBROUTINE

Aceste proceduri sau subprograme asigură transmiterea mai multor rezultate programului principal. De asemenea pot fi folosite pentru efectuarea unor operații de rutină - citire/scriere de exemplu - fără a transmite valori programului principal.

Forma generală a unui astfel de subprogram este:

```
SUBROUTINE nume (a1,a2,...,an)
[Instrucțiuni de specificare]
Instrucțiuni executabile (cel puțin una)
RETURN (cel puțin una)
END
```

în care:

SUBROUTINE - cuvânt rezervat care arată că ceea ce urmează este o procedură de tip subrutină;

nume - numele simbolic al subrutinei;

$a_1, \dots, a_n$ - parametri formali care pot fi variabile, tablo-uri, nume de subprograme, asteriscuri (caracter ce indică un punct de întoarcere în programul principal). Acești parametri nu pot să apară într-o instrucțiune COMMON, EQUIVALENCE, DATA. De notat că parametrii formali se referă atât la valori care se transmit spre subrutină cât și la valori ce urmează a fi transmise spre programul principal în momentul utilizării, sau puncte de revenire în programul principal.

Într-o subrutină nu pot să apară instrucțiunile BLOCK DATA, FUNCTION, SUBROUTINE.

Apelarea subrutinei se face prin instrucțiunea CALL cu forma generală:

$CALL \text{ nume}(p_1, p_2, \dots, p_n)$

în care:

CALL - cuvânt cheie cu sensul de „CHEAMA”;

nume - numele subrutinei;

$p_1, \dots, p_n$ - parametrii actuali sau efectivi care pot fi constante, variabile, tablouri, nume de subprograme (FUNCTION sau SUBROUTINE).

- Et (et = etichetă) dacă parametrul formal corespunzător este $\neq$.

Dacă parametrii efectivi sînt nume de subprograme, acestea trebuie declarate prin instrucțiunea EXTERNAL.

Parametrii efectivi trebuie să coincidă în număr, tip și ordine cu parametrii formali.

Revenirea în programul principal la instrucțiunea imediat următoare instrucțiunii CALL se face în momentul apariției în subrutină a instrucțiunii RETURN (v.fig.9.1)

Exemplu

Subrutina SUMA (X,N,SX)

```

SUBROUTINE SUMA (X,N,SX)
  DIMENSION X (N)
  SX = 0
  DO 1 I = 1,N
1  SX = SX + X (I)
  RETURN
END
    
```


Program principal

```

|-----|
|-----|
|-----|
|-----|
| CALL SUMA (Y,N,SY)
| CALL SUMA (Z,N,SZ)
|-----|
|-----|
|-----|

```

Subrutina SUMA rezolvă suma unui șir de valori, care are numele X, are dimensiunea N și a cărei sumă se numește SX. În programul apelant apare prima instrucțiune CALL, care cere rezolvarea prin intermediul procedurii SUMA a sumei șirului Y, de dimensiune N și a cărei sumă se numește SY.

După calculul sumei șirului Y de către procedura SUMA, se revine în programul principal la instrucțiunea imediat următoare, care este tot o instrucțiune CALL care indică același lucru, dar pentru șirul cu numele Z, cu dimensiunea N, a cărei sumă se numește SZ.

Observații

Procedurile de tip subrutină ca și cele tip FUNCTION sînt singurele programe care acceptă declarații de dimensiune în care indicii tablourilor sînt variabile de tip întreg. În acest caz dimensiunile tablourilor apar ca parametri în linie de definiție a procedurii.

Această observație decurge din posibilitatea utilizării aceluși proceduri, de programe ce prelucrează tablouri de dimensiuni diferite. Se asigură în acest fel un grad înalt de generalitate a procedurilor.

9.7 Instrucțiunea RETURN

Are forma generală:

```

|-----|
| [et] | RETURN
|-----|

```


sau

[et] | RETURN i

în care:

et - etichetă opțională;

RETURN - cuvânt cheie cu sensul de " ÎNTOARCERE ";

i - constantă întreagă $i \geq 1$.

Instrucțiunea RETURN indică sfârșitul logic al unei proceduri subrutină sau FUNCTION asigurând revenirea la programul principal, în instrucțiunea apelantă în cazul procedurii FUNCTION și în instrucțiunea imediat următoare instrucțiunii CALL în cazul procedurii subrutină.

Instrucțiunea RETURN i se folosește numai pentru proceduri subrutină. Asigură revenirea la programul principal în instrucțiunea cu etichetă scrisă ca parametru efectiv, a instrucțiunii CALL, corespunzător celui de-al i-lea asterisc - parametru formal din linia de definiție a procedurii subrutină.

Exemplu

Program principal

	CALL FØR (0, &1, &2, &3)
1	_____
2	_____
3	_____

Subrutina FØR

	SUBROUTINE FØR (A, *, *, *)

	IF(A) 11,12,13;
11	RETURN 1

12	RETURN 2

13	RETURN 3

Din exemplul de mai sus se observă că revenirea la programul principal din procedura FØR se face în instrucțiunea 1 dacă $A < 0$, în instrucțiunea 2 dacă $A = 0$ și în instrucțiunea 3 dacă $A > 0$ deoarece parametrul efectiv &1 corespunde primului aste-

riesc din linia de definiție a procedurii IPR ș.a.m.d.

9.8 Instrucțiunea ENTRY
are forma generală:

ENTRY i(f₁,f₂,...,f_n)

în care:

ENTRY - cuvânt cheie cu sensul de "INTRARE, ACCES";

i - identificator sau nume al punctului de intrare;

f₁...f_n - parametri formali n > 0.

Este o instrucțiune neexecutabilă care definește un punct de intrare într-o procedură subrutină sau FUNCTION, altul decât linia de definiție a acesteia.

În această situație intrarea în procedură se face tot prin instrucțiunea CALL pentru subrutină sau prin referire la procedura FUNCTION folosindu-se parametri efectivi care coincid ca număr, tip și ordine cu parametri formali din instrucțiunea ENTRY corespunzătoare.

Exemplu

Program principal

```

10  -----
    CALL MEDIA (X, Y, 2)
    -----
20  CALL ENT (T, V)
    -----
30  CALL SOR (W)
    -----

```

Procedura MEDIA

```

SUBROUTINE MEDIA (A, B, C)
    -----
    ENTRY ENT (T1, V1)
    -----
    ENTRY SOR (W1)
    -----
END

```


În exemplul dat, instrucțiunea 10 va determina executarea subrutinei MEDIA în întregime, în timp ce instrucțiunea 20 va determina executarea subrutinei MEDIA începînd de la instrucțiunea ENTRY ENT (T1,V1), iar instrucțiunea 30 va determina executarea procedurii MEDIA de la instrucțiunea ENTRY SØR (W1).

9.9 Instrucțiunea EXTERNAL

Are forma generală:

EXTERNAL $n_1, n_2, \dots, n_m$

în care:

EXTERNAL - cuvînt cheie cu sensul de „EXTERN”;

$n_1, \dots, n_m$ - nume de proceduri subrutine sau funcție, care sînt parametri actuali ai altor subprograme.

Instrucțiunea EXTERNAL este o instrucțiune de specificare și trebuie să precedă orice instrucțiune executabilă în programul principal.

Exemplu

Programul principal

```
-----
EXTERNAL RØD
-----
CALL VAR (A,RØD)
-----
```

Procedură

```

SUBROUTINE VAR (R,X)
-----
IF(R) 2,2,3
-----
2 | A=R*2+X*3
  | RETURN
  | END
```

Numele procedurii RØD apare cu parametru actual în instrucțiunea de apelare a procedurii VAR. În momentul executării procedurii VAR procedura RØD este chemată și executată numai cînd variabila R ≤ 0.

Exemplu

Program principal

```
CALL TES (X,Y,S(T,V))
```

Procedură

```
SUBROUTINE TES (X1,Y1,Z1)
```

```
RETURN
```

```
END
```

În acest exemplu nu este nevoie de instrucțiunea FUNCTION. S rezultatul activității acesteia devine cel de al 3-lea parametru actual în instrucțiunea CALL pentru apelarea subrutinei TES.

9.10. Instrucțiuni de organizare a memoriei

9.10.1 Instrucțiunea COMMON

Are forma generală:

$$\text{COMMON } /n_1/e_1, /n_2/e_2, \dots, /n_n/e_n$$

în care:

COMMON - cuvînt cheie în sensul de „COMUN”;

$n_1, n_2, \dots, n_n$ - identificatori și sînt nume de blocuri comune (sau etichete de blocuri comune).

Pot să se repete sau să se lipească. Dacă lipsește numele, respectivul bloc se numește bloc comun blank, între barele ce marchează numele blocului se lasă spațiu liber, iar dacă este pe prima poziție pot să nu apară respectivele bare.

$e_1, e_2, \dots, e_n$ - liste de numere ale variabilelor din care se constituie blocurile comune. Ele pot fi variabile simple, indexate și tablouri.

Instrucțiunea COMMON este folosită pentru a defini un domeniu de memorie la care se poate referi atât un program principal cît și unul sau mai multe subprograme, specificînd numele variabilor și tablourilor care vor fi plasate în domeniu.

Deci variabilele și tablourile care apar într-un program principal sau într-un subprogram pot fi făcute să ocupe aceleasi adrese în memorie cu variabilele sau tablourile din alte programe.

Poate fi folosit și pentru a transfera implicit argumente-

le între programul principal și un subprogram. Instrucțiunea COMMON permite lucrul la același program a unui număr de programatori care pot folosi domenii de memorie comune. Fiind o instrucțiune de specificare se va plasa după instrucțiunile de tip și dimensiune, înaintea oricărei instrucțiuni executabile.

Exemplu

Program principal

```

-----
-----
-----
COMMON/X1/ A,B,C (20)
-----
CALL XAL (P)
-----

```

Subrutina

```

SUBROUTINE XAL (R)
-----
-----
COMMON/X1/ T,V,X (20)
-----
-----

```

În exemplul dat, blocul comun X1 din instrucțiunea COMMON conține variabilele A,B și tabelul C cu 20 elemente în ordinea indicată.

Instrucțiunea COMMON din subrutina XAL face variabilele T, V, Y(1)-Y(20) să ocupe același spațiu de memorie (blocul comun X1) ca și variabilele A,B și tabelul C(20), așa cum se observă mai jos.

Bloc comun X1

A	B	C(1)	- - -	C(20)
T	V	Y(1)	- - -	Y(20)

Program principal
subrutină XAL

Observații

Numărul și tipul variabilelor din lista unui bloc etichetat trebuie să fie identice în toate unitățile de program care folosesc blocul comun. Regula nu este variabilă pentru bloc comun blanc.

9.10.2 Instrucțiunea EQUIVALENCE

Forma generală a instrucțiunii este:

EQUIVALENCE(e₁), (e₂), ..., (e_n)

în care:

EQUIVALENCE - cuvânt cheie cu sensul de "ECHIVALENT"
l₁, l₂, ..., l_n - listele de variabile, tablouri care se declară echivalente.

Instrucțiunea are rolul de a face ca aceeași locație de memorie să fie alocată la două sau mai multe variabile din același program, care apar la momente diferite.]

Dacă logica programului o permite se poate reduce spațiul de memorie ocupat făcând ca aceleași blocuri de memorie să fie ocupate de variabile de același tip, nu de tip diferit. Deci echivalența se referă la ocuparea aceleiași locații de memorie și la echivalență valorică.

Exemplu

DIMENSION A(25,5), B(125)

EQUIVALENCE (A,B), (V1,V2)

În exemplul de mai sus se alocă aceeași zonă de memorie tablourilor A și B, cât și variabilelor V1 și V2. Aceasta înseamnă că valorile tabloului A au fost folosite și nu vor mai fi apelate în momentul apariției apelării elementelor tabloului B. Același lucru se întâmplă și cu variabilele V1 și V2. Se realizează în acest mod o economie apreciabilă de memorie așa cum se observă în schema de mai jos.

A(1,1)	A(2,1)	- - -	A(25,5)	V1
B(1)	B(2)	- - -	B(125)	V2

Exemplu

DIMENSION A(3,3), B(9)

EQUIVALENCE (A(3,1), B(1))

În exemplul de mai sus echivalarea se face începând de la anume elemente ale tablourilor. Aceasta atrage după sine echivalarea și pentru elementele tablourilor ce urmează, dar nu pentru cele dinainte, așa cum se observă în schema de mai jos.

A(1,1)	A(2,1)	A(3,1)	A(1,2)	A(2,2)	A(3,2)	A(1,3)	A(2,3)	A(3,3)
B(1)	B(2)	B(3)	B(4)	B(5)	B(6)	B(7)	B(8)	B(9)

9.10.3 Procedura BLOCK

Procedura BLOCK DATA se folosește pentru utilizarea variabilelor care se află într-un bloc comun etichetat operație, pe care instrucțiunea DATA nu poate face.

Această procedură nu poate conține decât următoarele instrucțiuni: DATA, COMMON, DIMENSION, EQUIVALENCE, declarații de tip, ultimele trei în mod opțional. Nu poate conține instrucțiuni

uni executabile.

Forma generală este:

```
BLOCK DATA  
[Decl. tip]  
[DIMENSION]  
[EQUIVALENCE]  
COMMON  
DATA  
END
```

Exemplu

```
BLOCK DATA  
COMPLEX A,B  
COMMON /I/ A, XI /RER/ I, B  
DATA I, XI /25, 4.2/ V /1., - 3.5/
```

Se observă că este obligatorie listarea tuturor variabilelor în instrucțiunea COMMON, chiar dacă ele nu sînt utilizate, așa cum se întîmplă cu variabila complexă A.

CAPITOLUL 10.

UTILIZAREA BIBLIOTECILOR MATHLIB SI STATIST.

10.1. Prezentarea bibliotecilor MATHLIB si STATIST.

Calculatorul FELIX C-256 este prevăzut cu biblioteci de programe ce rezolvă o serie de probleme de analiză numerică-biblioteca MATHLIB, și de statistică-biblioteca STATIST.

Modul de utilizare a acestor programe este prezentat pe larg în manualul de utilizare a bibliotecilor respective în ce privește numele subprogramului, algoritmul utilizat, numele și semnificația parametrilor fictivi, modul de apelare al subprogramului. Cele două biblioteci sînt furnizate în forma sursă (limbaj FORTRAN) și în forma IMT. În funcție de forma în care se dorește utilizarea programului, apelarea se face diferit. În aceste biblioteci programele sînt de tip subrutină. Pentru chemarea subrutinei în programul principal trebuie folosită instrucțiunea corespunzătoare. Deoarece aceste subprograme nu efectuează operații de intrare/ieșire transferul datelor inițiale și a rezultatelor se realizează cu ajutorul argumentelor formale.

Tablourile utilizate în subprograme sînt monodimensionale, de dimensiuni ajustabile pentru a nu ocupa spații de memorie inutile. În programul principal aceste tablouri vor fi reprezentate pe coloane.

Pentru fiecare subprogram se indică formula de calcul ce permite determinarea numărului de locații ocupate de subprograme. Pentru fiecare subprogram există două variante care prelucresc datele în simplă precizie, sau în dublă precizie. Această calitate este specificată de caracterul "D", plasat în prima poziție a numelui subprogramului respectiv.

Biblioteca MATHLIB conține subprograme privind operații elementare cu matrici, calculul elementelor proprii ale unor matrici, rezolvarea unui sistem de ecuații, calculul integralelor definite etc.

Biblioteca STATIST cu volume STATIST1 și STATIST 2 conține subprograme de interes general și de previziune pe termen scurt, generarea de numere uniforme repartizate pe intervalul (0,1), generarea de variabile aleatoare uniforme repartizate pe intervalul

(a,b), generarea unei variabile aleatoare normale și după alte legi de distribuție, testul Student, testul Kolmogorov, folosirea celor mai mici pătrate la legăturile liniare, regresie polinomială prin metoda polinoamelor lui Cebîșev etc.

În continuare se vor prezenta câteva din subprogramele bibliotecilor MATHLIB și STATIST.

Subprogramul MRADD calculează adunarea a două matrici de tip real de dimensiuni identice.

Subprogramul se apelează cu instrucțiunea:

CALL MRADD (A,B,R,N,M)

în care:

A-numele tabloului care conține prima matrice

B-numele tabloului care conține matricea a doua

R-numele tabloului care conține matricea suma

N-numărul de linii al matricelor A,B,R

M-numărul de coloane al matricelor A,B,R

Subprogramul MRDIF calculează diferența a două matrici de tip real de aceleași dimensiuni. Matricea diferență este

$$R = A - B$$

Subprogramul se apelează cu instrucțiunea

CALL MRDIF (A,B,R,N,M)

în care parametrii formali au aceleași semnificații ca mai sus.

Subprogramul MRMUL calculează produsul a două matrici de tip real.

Apelarea subprogramului se face cu instrucțiunea:

CALL MRMUL (A,B,R,N,M,L)

Subprogramul RESOL rezolvă un sistem liniar de ecuații prin metoda lui Gauss.

Subprogramul este apelat cu instrucțiunea:

CALL RESOL (A,B,N,KOD,EPS)

în care:

A-tablou unidimensional în care se dispune pe coloane matricea coeficienților sistemului;

B-tablou de dimensiune N, în care inițial sînt aranjate valorile termenilor liberi din membrul doi, iar după efectuarea calculelor conține soluția X a sistemului;

N-dimensiunea sistemului, egală cu numărul de ecuații și de necunoscute;

KOD-codul de eroare, care ar urma să se tipărească în programul principal: pentru KOD=0 soluție normală, pentru KOD=1 matricea A este singulară (DET=0);

EPS= valoare limită sub care un element pivot (de pe diagonala principală) se consideră nul.

Subprogramul INROM calculează valoarea integralei definite $\int_{x_1}^{x_2} f(x) dx$ prin metoda Romberg, funcția fiind definită pe intervalul (x_1, x_2) .

Subprogramul se apelează cu instrucțiunea:

CALL INROM (FON, X1, X2, ITER, EPS, KOD, B, R)

în care:

FON-numele funcției dată de utilizator

x₁-limita inferioară a intervalului de integrare

x₂-limita superioară a intervalului de integrare

ITER-numărul maxim de iterații permise

EPS-precizia impusă

KOD-cod de eroare, care poate fi:

KOD=0 soluție normală

KOD=1 precizia dorită nu este obținută după numărul de iterații impus (ITER)

B-tablou de lucru cu dimensiunea cel puțin egală cu ITER+1

R-valoarea integralei.

În programul principal instrucțiunea apelantă, CALL va fi precedată de o declarație:

EXTERNAL FON

Funcția de integrat va fi furnizată programului prin intermediul unui subprogram FUNCTION:

FUNCTION FON (X)

în care:

FON-expresia algebrică de sub semnul integralei

FUNCTION FON (X)

FON=expresia algebrică a lui f(x)

RETURN

END

Subprogramul INSIM calculează valoarea unei integrale definite $\int_{x_1}^{x_2} f(x)dx$ prin metoda lui Simpson. Se apelează cu instrucțiunea:

CALL INSIM (FON,X1,X2,EPS,ITER,R1,R,KOD)

în care:

parametri formali au aceeași semnificație ca și la subprogramul INROM

R1=valoarea integralei calculate înaintea ultimei iterații, iar codul de eroare KOD are valorile:

KOD=0-soluție normală

KOD=1-cazul când $x_1=x_2$

KOD=2-când precizia este inferioară valorii de 10^{-9}

KOD=3-când numărul de iterații este inferior lui 2

KOD=4-cazul când precizia EPS nu este atinsă după ITER iterații.

Si în acest caz instrucțiunea de chemare a subprogramului va fi precedată de instrucțiunea EXTERNAL FON, iar funcția de integrat va fi, de asemenea, furnizată de utilizator printr-o funcție subprogram FUNCTION FON(X).

Din biblioteca STATIST se prezintă subprogramul TESTT care calculează variabila STUDENT a unei mulțimi de valori.

Subprogramul se apelează cu instrucțiunea:

CALL TESTT (A,NA,B,NB,IOPT,NDL,RES)

în care:

A-tablou de dimensiune NA ce conține primul eșantion de valori

B-tablou de dimensiune NB ce conține al doilea eșantion de valori

IOPT-codul ce indică opțiunea aleasă de utilizator privind mulțimea B și dimensiunea NA

RES=valoarea calculată a variabilei STUDENT

NDL-numărul de grade de libertate asociate acestei variabile.

Apelarea unui subprogram dintr-o bibliotecă MATHLIB sau STATLIB se face folosind cartela FETCHS cu formula generală:

1 3
* FETCHS LN:nume1,DV:perif,FN:nume2,VN:nr1,GN:nr2,UN:nr3

în care:

nume 1-numele bibliotecii apelate
nume 2-numele cărții din bibliotecă
perif-perifericul pe care se află memorată biblioteca.
nr.1-numărul de versiune al bibliotecii
nr.2-numărul de generare al bibliotecii
nr.3-numărul de actualizare

în cazul utilizării compelatorului FLAG, plasată după cartela
% COMPILE FLAG

În cazul utilizării compilatorului FORTRAN se folosește
cartela LIB cu forma generală:

1 11
. LIB LN:nume 1,DV:perif,VN:nr 1,GN:nr 2

notațiile avînd aceleași semnificații ca și mai sus. Car-
tela LIB va fi plasată imediat după cartela JOB.

10.2. Exerciții.

1) Să se calculeze matricea produs a matricelor A(3,4) și
B(4,6).

Se va folosi subprogramul MKUL care efectuează această în-
mulțire, fiecare termen al matricei rezultat C este egal cu:

$$C_{ij} = \sum_{k=1}^n \sum_{l=1}^l \sum_{j=1}^m a_{lk} b_{kj} \quad \begin{matrix} i=1, n; j=1, \\ k=1, l \end{matrix}$$

Produsul are sens numai dacă numărul coloanelor matricei
A este egal cu numărul liniilor matricei B.

Rezultatele rulării programului este prezentat pe pagina

2) Să se afle valoarea integralei definite:

$$F(x) = \int_{x_1}^{x_2} 18x^3 - 27x^2 + x + 4 \, dx$$

Se utilizează pentru rezolvarea acestei integrale metoda
lui Romberg, materializată în subprogramul cu numele INROM pentru
simpla precizie și DINROM pentru dubla precizie.

Rezultatul activității programului este prezentat la pag.

Se observă că s-a organizat o procedură FUNCTION FON pentru
funcția de sub semnul integralei.

Imprimanta cu programul si rezultatul activitatii
subprogramului MRMUL

```

1      C      INMULTIREA A DOUA MATRICI UTILIZIND SUBPROGRAMUL MRMUL
2      REAL MA,MB,MC
3      DIMENSION MA(3,4),MB(4,6),MC(3,6)
4      READ(105,1) MA,MB
5      1 FORMAT (12F4,2/24F5,2)
6      WRITE(108,2) MA,MB
7      2 FORMAT(30X,'MATRICEA MA      '/30X,12(1H*)//3(27X,4(F4,2,1X)42//30
8      CX,'MATRICEA MB      '/30X,13(1H*)//4(24X,6(F3,2,1X)72)
9      WRITE(108,3)
10     3 FORMAT (30X,'MATRICEA REZULTAT      MC      '/30X,22(1H*)//)
11     CALL MRMUL(MA,MB,MC,3,4,6)
12     WRITE(108,4) MC
13     4 FORMAT (3(15X,6(F7,2,1X)/))
14     STOP
15     END

```

MATRICEA MA

41.24	91.42	01.20	10.47
05.24	70.91	20.20	5.91
34.20	57.23	20.14	23.31

MATRICEA MB

1.42	2.71	2.21	0.20	1.41	3.11
1.39	4.21	5.63	2.01	4.20	4.12
3.10	4.25	0.24	0.14	7.09	2.72
1.44	1.42	5.67	0.70	1.43	2.15

MATRICEA REZULTAT MC

000.01	1100.22	000.20	500.11	700.00	000.00
522.33	1077.85	777.13	091.07	1401.33	947.12
424.81	1027.33	814.44	472.14	1100.00	707.09



-141-

Programul și imprimanta cu rezultatul activității
subprogramului INROM

```

1 C PROGRAMUL PRINCIPAL
2 C
3 READ(105,1)X1,X2,ITER,EPS
4 1 FORMAT(2F2.1,I1,F4.2)
5 EXTERNAL FON
6 CALL INROM(FON,X1,X2,ITER,EPS,KOD,B,R)
7 WRITE(108,2) R,KOD
8 2 FORMAT(30X,'VALOAREA INTEGRALEI F(X) ESTE'//30X,29(1H=)//35X,'F(X)
9 C=' ,G14.7//40X,'KOD=' ,I2)
10 STOP
11 END

```

INT1

26/02/76 00,10.58

```

1 C SUBPROGRAMUL FUCTION
2 FUNCTION FON(X)
3 FON=SQRT(18*X**3-27*X**2+X+4)
4 RETURN
5 END

```

VALOAREA INTEGRALEI F(X) ESTE

=====

F(X)= 98,35547

KOD= 1

CAPITOLUL 11

DETECTAREA ERORILOR UNUI PROGRAM

În practica programării calculatoarelor apar frecvent diferite greșeli ce aparțin programatorului sau accidente în funcționarea unui ansamblu ce constituie sistemul de calcul (calculatorul), lucru ce face ca programul întocmit să ducă fie la rezultate eronate, fie să nu poată fi executat.

Pentru evitarea unor astfel de situații sistemul este prevăzut cu subprograme specializate detectoare de erori. Capitolul de față prezintă problema tratării erorilor semnalate de către sistem prin afișarea unor mesaje în clar sau codificate. Programatorul ce depanează un program trebuie să plece întotdeauna de la ideea că CEL CARE A GRESIT ESTE EL ȘI NU SISTEMUL DE CALCUL.

Apariția erorilor se datorează mai multor cauze:

- cunoașterea insuficientă a sintaxei limbajului de programare;
- scrierea incorectă sau neciteată a programului pe foaia de programare;
- scrierea incorectă a cartelelor de comandă;
- ignorarea restricțiilor impuse de limbaj;
- utilizarea incorectă a procedurilor.

Subliniem faptul că de cele mai multe ori, operatoarea de la mașina de perforat cartele, nu cunoaște limbajul în care este scris programul și deci nu ne putem aștepta ca ea să corecteze o eventuală greșeală; pe de altă parte nici nu are timpul necesar pentru a-și pune problema ce anume a vrut să scrie în locul respectiv programatorul. Deci, repetăm, programatorul este obligat să transcrie programul pe formularul de programare cât se poate de citet, evitând în acest fel o potențială sursă de erori, astfel încât să nu dea naștere la dubii. Astfel, litera I scrisă necitet poate să fie luată drept l sau semnul împărțirii (/), ceea ce este cu totul altceva. În felul acesta pot apare gre-

seli grave pe suportul de intrare (cartela), care amplifică numărul erorilor de sintaxă pe suportul de ieșire (imprimantă).

ERORILE UNUI PROGRAM

La introducerea unui program nou în calculator pot fi semnalate următoarele tipuri de erori:

1. Erori semnalate de programul care tratează cartelele de comandă, referitoare la punerea în lucru a programului și a compilatorului.

Aceste erori duc la abandonarea lucrării până la apariția unui nou JOB și se datoresc nerespectării sintaxei cartelelor de comandă sau utilizării altor cuvinte cheie decât cele impuse de documentația sistemului de operare.

2. Erori de compilare

Sînt detectate de către compilatorul FORTRAN la analiza textului sursă. Indiferent de opțiunile ce figurează pe cartela COMPILE, erorile detectate sînt grupate și afișate la imprimantă la sfîrșitul compilării după eventuale liste și tabele cerute. Erorile detectate la compilare sînt împărțite pe patru nivele de gravitate:

- nivelul 1 - erori banale (pot fi sau nu corectate); de exemplu, într-o expresie aritmetică lipsește o paranteză dreaptă. În acest caz compilatorul presupune că aceasta se găsește la sfîrșitul expresiei;

- nivelul 2 - erori nu prea grave; de exemplu, într-un program se face o dublă definire

```
DIMENSION A (10)
```

```
DIMENSION A (10)
```

În acest caz compilatorul păstrează numai prima definiție.

- nivelul 3 - erori grave; nu este generat nimic pentru faza corespunzătoare; de exemplu, într-o referire la un element de tablou lipsește indice

```
DIMENSION A (10)
```

```
A=B+C
```


- nivelul 4 - erori foarte grave, erori ce opresc întotdeauna compilarea.

De exemplu, nici o instrucțiune executabilă în secvența de program:

```

COMPILE FORTRAN
DIMENSION A(10)
DATA A/10 = 0./
END
    
```

Erorile de nivel mai mic decât 4, pot fi ignorate de către compilator prin specificarea opțiunii ERR:4, din cartela COMPILE, dar nu există siguranța obținerii unor rezultate corecte. Orice eroare de nivel 4 întrerupe compilarea, interzicându-se editarea de legături, chiar dacă s-a introdus opțiunea amintită. Erorile detectate la compilare sînt împărțite în două categorii:

Erori detectate în timpul analizei sintactice

Pentru fiecare eroare detectată se afișează la imprimantă un mesaj de forma:

```

** LINIA nr.  TEXT INSTRUCTIUNE
   NIVEL nr.  MESAJ EXPLICATIV
    
```

Sub caracterul care a provocat eroarea sau care a permis detectarea ei, este imprimat caracterul I. În continuare apare tipărit nivelul de eroare și un mesaj explicativ.

Exemplu:

```

** LIGNE 20  FORMAT (26H)
               I
    
```

NIVEAU 3: DESCRIPTEUR DE ZONE ERRONE

Mesajele de eroare care apar la analiza sintactică sînt prezentate în anexa A.//7/

Erori detectate în timpul alocării de variabile

Acste erori privesc alocările propriu-zise de variabile, adică cele indicate de instrucțiunile COMMON, EQUIVALENCE, alocarea zonelor pentru FORMAT.

Exemplu: mărimea unui bloc COMMON depășește 64 KO

Avertisamente de erori

În afară de mesajul explicativ relativ la erorile detectate la compilare, uneori se afișează și avertismente de eroare. Nu sînt erori în sensul propriu-zis al cuvîntului, ele neafectînd conținutul programului și de aceea nu le este afectat nici un nivel.

AVERTISMENT : ETIQUETTE '2' DEFINIE, NON REFERENCE.

Această etichetă însoțește instrucțiunea `FORMAT` (este deci definită), dar nu apare în nici o instrucțiune de `I/I` (dacă nu este referită).

Lista erorilor de compilare se încheie cu afișarea gestiunii de erori care indică numărul erorilor și nivelul acestora.

Incidente în timpul compilării sau catalogării

Dacă în timpul compilării sau catalogării apar incidente - deosebit de grave care interzic continuarea compilării sau catalogării, aceste operații sînt abandonate și se tipărește pe ecranul de ieșire (imprimantă) un mesaj explicativ.

Erori semnalate de editorul de legături

Ca și în cazul compilării, erorile semnalate de editorul de legături sînt grupate pe 4 nivele. Cele mai frecvente erori sînt generate de:

- neconcordanța tipului parametrului formal al subprogramului din biblioteca sistemului cu parametrul actual;
- folosirea într-o instrucțiune `CALL` a unui nume de subprogram tip subrutină, altul decît cel declarat în linia de definire a acestuia;
- folosirea unor nume de subprograme diferite de cele din biblioteca sistemului;
- adresarea unor variabile care nu sînt declarate tablou și care nu fac parte din lista de variabile `READ` sau `WRITE`;
- sintaxa cartelei `THRE` este eronată.

Aceste erori, cît și altele de acest gen nu permit generarea în formă `IMT` a programului și ^{impun} deci oprirea programului în faza de linkeditare (`E-D-L`).

Erori la lansarea programului

În momentul în care se încarcă un program pentru a fi lansat în execuție, pot apărea diferite erori care se comunică sub forma unor mesaje. Prin conținutul lor aceste mesaje specifică lipsa unor resurse necesare încărcării și lansării programului, ca: lipsă de memorie, lipsă periferice, cartele relative la prelucrarea fișierului eronate.

Erori de execuție a programului

Erorile care apar în faza de execuție a programului sunt comunicate la sfârșitul acestei faze și sunt de două tipuri:

Erori detectate la operațiile de calcul

Aceste erori sunt afișate la imprimantă sub forma unui mesaj cu conținutul:

ERREUR PROGRAME xxxxxxxx

unde xxx este o succesiune de 4 cifre zecimale care indică codul erorii. În anexa C se prezintă toate erorile de acest tip și codurile lor /7/.

Tratarea acestor erori este ceva mai dificilă decât cele precedente, ele presupunând oarecare cunoștințe privind limbajul de asamblare, conținutul și grafica programului obiect și a vidajului de memorie, cunoștințe despre care se vorbește în lucrarea /7/. Cele mai frecvente erori de acest tip, care apar în calculele tehnico-științifice, sunt erorile specificate cu codurile 0101 și 0108. În ambele cazuri se semnalizează o depășire de capacitate pentru modul de exprimare a numerelor în binar, respectiv în virgulă flotantă (constante întregi sau reale).

Erori detectate în execuția unei operații de I/I (la blocurile de comandă a perifericelor)

Pentru aceste erori se comunică un mesaj de forma:

ERREUR E/S yy xxxxxx

unde: yy este codul erorii;

xxxxxx este adresa blocului de comandă asociat operației de I/I.

Această categorie de erori apare când:

- lipsește un aparat periferic;
- apar incidente ireparabile pe bandă sau disc la scriere sau consultare;

- depășirea numărului de linii specificat în cartela RUN.
In anexa D /7/ sînt prezentate erorile de acest gen și codurile respective.

Erorile sistemului de gestiune a fișierelor

Aceste erori sînt detectate de modulele SGF în timpul operațiilor asupra fișierelor și semnalate cu ajutorul unor mesaje de forma:

nn E G F dd cc mm....m

unde: nn - identificatorul de exploatare;

EGF - eroarea gestiune fișiere;

dd - codul care identifică originea erorii;

cc - codul erorii;

mm....m - succesiune de caractere literale care specifică eroarea.

Aceste erori se tratează verificînd:

- instrucțiunile de descriere (DEFINE FILE, USED FILE)
- instrucțiunile de prelucrare (READ, FIND, WRITE)
- concordanța dintre index din cartela DEFINE FILE și cartelele LABEL, ASSIGN. Sînt prezentate în anexa E /7/.

Prezentarea listei modului obiect

În faza de compilare, după efectuarea verificării sintactice, se execută "traducerea" programului sursă în limbaj cod mașină. Se generează instrucțiuni cod mașină corespunzătoare fiecărei instrucțiuni sursă. Totodată are loc atribuirea de adrese de memorie atît instrucțiunilor în cod mașină, cît și variabilelor care intră în lucru. În faza de editare a legăturilor are loc încorporarea de subprograme din biblioteca de subprograme (module a căror nume încep cu caracterele I%, F% și module SGF) care ajută la derularea corectă a programului în unitatea centrală. În plus se execută modificarea adreselor atribuite în faza de compilare cu lungimea zonei punctelor de intrare.

Programul obiect este afișat la imprimantă dacă pe cartela COMPILE apare opțiunea OBL. În ce privește gruparea informațiilor ce constituie programul obiect din punct de vedere grafic, lista se prezintă astfel: (vezi /7/).

a) pe verticală

1. Zona care specifică linia FORTRAN căreia îi corespunde

secvența de instrucțiuni cod mașină generate (notată pe imprimantă cu nr.1);

2. zona adreselor atașate instrucțiunilor generate, variabilelor și cuvintelor rezervate de compilator, începînd de la zero. Zona de adresă se prezintă sub forma de 2 octeți, (notată cu nr.2);

3. zona cuprinzînd conținutul cuvintelor rezervate de compilator, instrucțiuni cod mașină și echivalentul lor în limbaj ASSIRIS - format extern (notată la imprimantă cu nr.3).

b. pe orizontală

1. Zona de informații folosite exclusiv de Sistemul de gestiune a fișierelor (SGF), care cuprinde tabele de descriere a fișierelor (zonă notată cu I pe imprimantă de la zero);

2. Zona de informații rezervată de compilator instrucțiunii DATA, NAMELIST, FORMAT (zona II pe imprimantă);

3. Zona instrucțiunilor cod mașină corespunzătoare instrucțiunilor FORTRAN în format intern (cod mașină) și echivalentul lor în limbaj ASSIRIS (limbaj simbolic), (zona III pe imprimantă).

Prezentarea imaginii programului în memoria calculatorului

Imaginea memoriei afectată execuției programului se prezintă astfel:

1. Zona informațiilor generale, care ajută la gestionarea execuției programului sub controlul monitorului și care conține: nume JOB, zona FILE, ASSIGN, LABEL, tabela de segmente (dacă programul este segmentat), (zonă notată cu mesaje în clar la imprimantă /7/.

2. Zona de prelucrare conține segmentul în curs de execuție dacă programul este segmentat, sau tot programul dacă nu este segmentat. Zona de prelucrare începe de la adresa de bază conținută în registrul 14 și are următoarea structură:

- zona punctelor de intrare - toate cuvintele care au la codul funcției "00" și încep cu "6" (vezi instrucțiunea FORTRAN);

- modulul obiect 1;
- modulul obiect 2;
- modulul obiect 3;
- modulul obiect 4.

3. Zona de opțiuni

Vidajul de memorie propriu-zis se prezintă la imprimantă astfel:

- două coloane de adrese din cifre 1 octeți (notate cu A și B la imprimantă);

- opt coloane de cuvinte (patru octeți), care conțin date sau instrucțiuni așezate pe primele patru coloane după coloana A, următoarele patru coloane după coloana B.

Vidaful de memorie este precedat întotdeauna, (de fapt face parte din el), de conținutul cuvintelor care reprezintă starea programului și de conținutul celor 16 registre.

Primul cuvânt al dublului registru de stare a programului reprezintă adresa absolută a instrucțiunii ce urmează după instrucțiunea care a provocat eroarea (subliniat pe imprimanta de la pagina 92 /7/).

Conținutul registrului 14 reprezintă baza segmentului în curs de execuție. Conținutul registrului 15 reprezintă adresa de la început a tabelului de segmente a programului care conține segmentele listate de EDI.

Dacă un segment este încărcat respectivul cuvânt începe cu "00" și conține adresa unde a fost încărcat segmentul s, dacă un segment nu a fost încărcat, respectivul cuvânt începe cu "80".

BIBLIOGRAFIE

1. Baltac V.s.a. **EBLIX C-256 structura și programarea calcula-**
latorului. Ed. Tehnică-București 1974
2. Costake N.s.a. **Fortran vol I și vol II**
Editura Tehnică-București 1971
3. Gruța C.s.a. **Programarea la calculatorul EBLIX C-256**
Editura științifică 1973
4. Cazacu C.s.a. **Programarea în limbajul FORTRAN**
Editura Junimea - Iași - 1978
5. Ciongradi Camelia **Introducere în programare Lito. IPI 1982**
6. Ciocoiu Mihai s.a. **Practica programării în FORTRAN**
Lito I.P. Iași 1974
7. Ciocoiu Mihai s.a. **Practica programării în FORTRAN p. II-a**
Lito I.P. Iași 1976
8. Ciocoiu Mihai s.a. **Programare în limbajul FORTRAN**
Lito I.P. Iași 1981
9. Chorafas D.N. **Traite des ordinateurs Ed. Hermann Paris 1960**
10. Dances I.s.a. **Programarea calculatoarelor numerice**
Editura Dacia - Cluj - 1973
11. Dimo P. **FORTAN IV. Editura Didactică și Pedagogică**
București 1971
12. Dora W.S.s.a. **Metode numerice cu programe în FORTRAN IV**
Editura Tehnică-București 1976
13. Foraythe A.s.a. **Programacion FORTRAN**
Editura Limusa Mexico 1976
14. Grigoras Ghe.s.a. **Culegere de probleme Lito- Universitatea**
Al.I.Cuza Iași 1977
15. Georgescu H.s.a. **Programe în limbajul FORTRAN**
Editura Albatros- București 1976
16. Livovchi L. **Scheme logice Ed. Tehnică București 1980**
17. Livovchi L. **Bazele informaticii**
Editura Albatros București 1976

18. Ledley R.S. Programming and utilizing digital computers McGraw New York 1962 Book Co.
19. Liascenko N.A.ş.a. Osnovi programirovaniia Kiev "Radianska şkola" 1979
20. Larionescu S. Calculatoare mici şi mari Lito I.C.Bucureşti 1972
21. Lamprecht G. Einführung in die Programmiersprache FORTRAN Vieweg Verlag Braunschweig 1973
22. Moldovan G. Scheme logice şi programe FORTRAN Ed.Didactică şi pedagogică Bucureşti 1978
23. Niculescu St- Iniţiere în FORTRAN Editura Tehnică Bucureşti 1972
24. Niculescu St. FORTRAN iniţiere în programarea structurală Editura Tehnică Bucureşti 1979
25. Niculescu S. Algoritmi - Editura Tehnică Bucureşti 1981
26. Poată N. Calculatoare automate şi programare Ed.Didactică şi Pedagogică Bucureşti 1973
27. Petruş O. Programarea în FORTRAN Junimea Iaşi 1980
28. Roşculeţ M.ş.a. Programarea şi utilizarea maşinilor de calcul - Ed.Didactică şi Pedagogică Bucureşti 1980
29. Toma M.ş.a. Metode numerice şi subrutine Editura Tehnică Bucureşti 1980
30. Văduva I.ş.a. Introducere în programarea automată Editura Didactică şi Pedagogică Bucureşti 1973
31. ### FORTRAN IV Manuel d' operation CII 1971 Franţa
32. ### FORTRAN IV Manuel d' utilisotion CII 1971 Franţa
33. ### Bibliothèque mathématique CII 1972 Franţa

CUPRINS

	pag.
Introducere	2
Cap. 1 - Bazele teoretice ale calculatoarelor	4
1.1. Scurt istoric al calculatoarelor	4
1.2. Clasificarea calculatoarelor	7
1.3. Bazele aritmetice ale calculatoarelor	8
1.3.1. Sisteme de numerație	8
1.3.2. Operații aritmetice	9
1.3.3. Conversiunea numerelor	13
1.4. Structura generală și funcționarea unui calcula- tor.	19
1.4.1. Generalități	19
1.4.2. Schema bloc a unui calculator	19
1.4.3. Descrierea unităților unui calculator	21
1.4.4. Principiile fundamentale de lucru ale calcula- torului	25
1.4.5. Modul de lucru al calculatorului numeric	26
1.5. Reprezentarea informațiilor în calculator	27
1.5.1. Informația alfanumerică	27
1.5.2. Informația numerică	27
Cap. 2 - Algoritmi și scheme logice	30
2.1. Etapele preliminare rezolvării unei probleme pe calculatorul electronic	30
2.2. Scheme logice de calcul	33
2.2.1. Scheme logice liniare	34
2.2.2. Scheme logice cu ramificații	35
2.2.3. Scheme logice cu cicluri	36
2.2.3.1. Scheme logice cu număr cunoscut de pași	36
2.2.3.2. Scheme logice cu număr necunoscut de pași	39

Cap. 3 - Elemente de bază ale limbajului FORTRAN	41
3.1. Generalități	41
3.2. Caractere FORTRAN	42
3.3. Constante	45
3.4. Variabile	49
3.5. Tablouri	50
3.6. Expresii	53
3.6.1. Expresii aritmetice	54
3.6.2. Expresii logice	59
3.7. Instrucțiunile limbajului FORTRAN	61
3.7.1. Clasificare	61
3.7.2. Structura unui program	63
3.7.3. Formularul de programare FORTRAN	63
3.7.4. Cartele de comandă	66
3.7.4.1. Fazele de rezolvare a unui program FORTRAN pe calculator	66
3.7.4.2. Cartele de comandă	66
3.7.5. Structura unui program FLAG	69
Cap. 4 - Instrucțiuni de intrare - ieșire	71
4.1. Forma generală a instrucțiunii I/I	71
4.2. Instrucțiunea de I/I standard	72
4.2.1. Instrucțiunea de intrare	72
4.2.2. Instrucțiunea de ieșire	73
4.3. Instrucțiunea FORMAT	74
4.3.1. Coduri de paginare	75
4.3.1.1. Codurile benzii pilit	75
4.3.1.2. Codul X	76
4.3.1.3. Codul /	76
4.3.1.4. Codul T	77
4.3.1.5. Codul H și apostrof	78
4.3.2. Coduri pentru transmiterea constantelor FORTRAN	81
4.3.2.1. Codul I	81
4.3.2.2. Coduri pentru transmiterea datelor reale	84
4.3.2.4. Codul L	90
4.3.2.5. Codul A	91
4.4. Alte forme ale instrucțiunilor I/I	93
4.4.1. Instrucțiunea de citire cu opțiuni	93
4.4.2. Instrucțiuni I/I FORTRAN II	94
4.4.3. Instrucțiuni I/I pentru tablouri	95
4.5. Relația dintre lista de intrare-ieșire și lista	

	de coduri FORMAT	96
4.6.	Utilizarea virgulei la scrierea datelor pe cartela 98	
4.7.	Alte instrucțiuni FORTRAN	99
4.7.1.	Instrucțiunea STOP	99
4.7.2.	Instrucțiunea PAUSE	100
4.7.3.	Instrucțiunea END	100
	Cap. 5 - Instrucțiunea de atribuire	101
5.1.	Instrucțiunea de atribuire aritmetică	101
5.2.	Instrucțiunea de atribuire logică	102
5.3.	Instrucțiunea DATA	103
	Cap. 6 - Instrucțiuni de specificare	104
6.1.	Instrucțiunea DIMENSION	104
6.2.	Instrucțiuni de specificare	105
6.2.1.	Instrucțiunea IMPLICIT	105
6.2.2.	Instrucțiunea de specificare explicită	106
6.2.3.	Instrucțiunea DOUBLE PRECISION	107
	Cap. 7 - Instrucțiuni de salt	108
7.1.	Instrucțiunea de salt necondiționat	108
7.1.1.	Instrucțiunea GOTO	108
7.1.2.	Instrucțiunea GOTO calculat	109
7.1.3.	Instrucțiunea GOTO impus	109
7.2.	Instrucțiunea de salt aritmetic	110
7.2.1.	Instrucțiunea IF aritmetic	110
7.2.2.	Instrucțiunea IF logic	112
	Cap. 8 - Instrucțiunea de ciclare	114
	Cap. 9 - Proceduri FORTRAN	120
9.1.	Generalități	120
9.2.	Funcții definite	120
9.3.	Funcții intrinseci	122
9.4.	Funcții externe uzuale	122
9.5.	Proceduri FUNCTION	123
9.6.	Procedura SUBROUTINE	125
9.7.	Instrucțiunea RETURN	127
9.8.	Instrucțiunea ENTRY	129
9.9.	Instrucțiunea EXTERNAL	130
9.10.	Instrucțiuni de organizare a memoriei	131
9.10.1.	Instrucțiunea COMMON	131
9.10.2.	Instrucțiunea EQUIVALENCE	132
9.10.3.	Procedura BLOC	

	Cap. 10 - Utilizarea bibliotecilor MATHLIB și	
	STATIST	135
10.1.	Prezentarea bibliotecilor MATHLIB și STATIST ..	135
	Cap. 11 - Detectarea erorilor unui program . .	142
	Bibliografie	150
	Cuprins	152